

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

Wintersemester 2024/25

## Wichtiger Hinweis

Diese Vortragsfolien dienen dazu, den Vortrag der/des Lehrenden zu unterstützen. Sie enthalten **nur einen Teil** der Lerninhalte. Wie groß dieser Teil ist, hängt von den konkreten Lerninhalten ab und kann von „praktisch alles“ bis „praktisch gar nichts“ schwanken. Diese Folien alleine sind daher **nicht für ein Selbststudium geeignet!**

Mindestens genauso wichtig wie die Vortragsfolien sind die Beispiel-Programme, Notizen und Tafelbilder, die vor Ihren Augen in den Vorlesungen erarbeitet werden. Diese sind im Git-Repository mit allen Zwischenschritten enthalten (<https://gitlab.cvh-server.de/pgerwinski/dbs>) und befinden sich in den zu den jeweiligen Kalenderdaten gehörenden Verzeichnissen (z. B. für den 2.10.2024 unter <https://gitlab.cvh-server.de/pgerwinski/dbs/tree/2024ws/20241002/>).

In jedem Fall: *Viel Erfolg!*

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

2. Oktober 2024

# Vorab: Online-Werkzeuge

- Diese Veranstaltung findet **in Präsenz** statt.  
Wir versuchen aber, auch eine Online-Teilnahme zu ermöglichen.
- **Mumble**: Seminarraum 2  
Fragen: Mikrofon einschalten oder über den Chat  
Umfragen: über den Chat – **auch während der Präsenz-Veranstaltung**
- **VNC**: Kanal 6, Passwort: `testcvh`  
Eigenen Bildschirm freigeben: VNC-Software oder Web-Interface *yesVNC*  
Eigenes Kamerabild übertragen: Web-Interface *CVH-Camera*
- Allgemeine Informationen: <https://www.cvh-server.de/online-werkzeuge/>
- Notfall-Schnellzugang: <https://www.cvh-server.de/virtuelle-raeume/>  
Seminarraum 2, VNC-Passwort: `testcvh`
- **Lehrmaterialien**: <https://gitlab.cvh-server.de/pgerwinski/dbs>

# Was sind Datenbanken?

Datenbank = Datenbestand + System, um darauf zuzugreifen

Anforderungen an Datenbank [\[Wikipedia\]](#):

- effizient
- widerspruchsfrei
- dauerhaft

# Was sind Datenbanken?

Datenbank = Datenbestand + System, um darauf zuzugreifen

Anforderungen an Datenbank [\[Wikipedia\]](#):

- effizient → Sortierung, Zugriffsalgorithmen
- widerspruchsfrei
- dauerhaft

# Was sind Datenbanken?

Datenbank = Datenbestand + System, um darauf zuzugreifen

Anforderungen an Datenbank [\[Wikipedia\]](#):

- effizient → Sortierung, Zugriffsalgorithmen
- widerspruchsfrei → automatische Konsistenzprüfungen
- dauerhaft

# Was sind Datenbanken?

Datenbank = Datenbestand + System, um darauf zuzugreifen

Anforderungen an Datenbank [\[Wikipedia\]](#):

- effizient → Sortierung, Zugriffsalgorithmen
- widerspruchsfrei → automatische Konsistenzprüfungen
- dauerhaft → Backup, Ausfallsicherheit



# Was sind Datenbanken?

## Datenbank ohne Computer



Bildquelle: [https://commons.wikimedia.org/wiki/File:A\\_Day\\_in\\_the\\_Life\\_of\\_a\\_Wartime\\_Housewife-\\_Everyday\\_Life\\_in\\_London,\\_England,\\_1941\\_D2379.jpg](https://commons.wikimedia.org/wiki/File:A_Day_in_the_Life_of_a_Wartime_Housewife-_Everyday_Life_in_London,_England,_1941_D2379.jpg)

# Was sind Datenbanken?

## Datenbank ohne Computer

- menschliche Anwesenheit erforderlich



Bildquelle: [https://commons.wikimedia.org/wiki/File:A\\_Day\\_in\\_the\\_Life\\_of\\_a\\_Wartime\\_Housewife-\\_Everyday\\_Life\\_in\\_London,\\_England,\\_1941\\_D2379.jpg](https://commons.wikimedia.org/wiki/File:A_Day_in_the_Life_of_a_Wartime_Housewife-_Everyday_Life_in_London,_England,_1941_D2379.jpg)

# Was sind Datenbanken?

## Datenbank ohne Computer

- menschliche Anwesenheit erforderlich
- sorgfältig geordnet für effizienten Zugriff  
→ sorgfältiger Umgang erforderlich,  
um Ordnung zu erhalten



Bildquelle: [https://commons.wikimedia.org/wiki/File:A\\_Day\\_in\\_the\\_Life\\_of\\_a\\_Wartime\\_Housewife-\\_Everyday\\_Life\\_in\\_London,\\_England,\\_1941\\_D2379.jpg](https://commons.wikimedia.org/wiki/File:A_Day_in_the_Life_of_a_Wartime_Housewife-_Everyday_Life_in_London,_England,_1941_D2379.jpg)

# Was sind Datenbanken?

## Datenbank ohne Computer

- menschliche Anwesenheit erforderlich
- sorgfältig geordnet für effizienten Zugriff  
→ sorgfältiger Umgang erforderlich, um Ordnung zu erhalten
- Jeweils 1 Person kann an 1 Papier arbeiten.  
→ Hardware-Unterstützung für Konsistenz



Bildquelle: [https://commons.wikimedia.org/wiki/File:A\\_Day\\_in\\_the\\_Life\\_of\\_a\\_Wartime\\_Housewife-\\_Everyday\\_Life\\_in\\_London,\\_England,\\_1941\\_D2379.jpg](https://commons.wikimedia.org/wiki/File:A_Day_in_the_Life_of_a_Wartime_Housewife-_Everyday_Life_in_London,_England,_1941_D2379.jpg)

# Was sind Datenbanken?

## Datenbank ohne Computer

- menschliche Anwesenheit erforderlich
- sorgfältig geordnet für effizienten Zugriff  
→ sorgfältiger Umgang erforderlich, um Ordnung zu erhalten
- Jeweils 1 Person kann an 1 Papier arbeiten.  
→ Hardware-Unterstützung für Konsistenz
- Kopien sehr aufwendig  
→ sorgfältige Archivierung erforderlich



Bildquelle: [https://commons.wikimedia.org/wiki/File:A\\_Day\\_in\\_the\\_Life\\_of\\_a\\_Wartime\\_Housewife-\\_Everyday\\_Life\\_in\\_London,\\_England,\\_1941\\_D2379.jpg](https://commons.wikimedia.org/wiki/File:A_Day_in_the_Life_of_a_Wartime_Housewife-_Everyday_Life_in_London,_England,_1941_D2379.jpg)

# Was sind Datenbanken?

## Zentraler Computer



Bildquelle: [https://commons.wikimedia.org/wiki/File:Ken\\_Thompson\\_\(sitting\)\\_and\\_Dennis\\_Ritchie\\_at\\_PDP-11\\_\(2876612463\).jpg](https://commons.wikimedia.org/wiki/File:Ken_Thompson_(sitting)_and_Dennis_Ritchie_at_PDP-11_(2876612463).jpg)

# Was sind Datenbanken?

## Zentraler Computer

- menschliche Anwesenheit erforderlich



Bildquelle: [https://commons.wikimedia.org/wiki/File:Ken\\_Thompson\\_\(sitting\)\\_and\\_Dennis\\_Ritchie\\_at\\_PDP-11\\_\(2876612463\).jpg](https://commons.wikimedia.org/wiki/File:Ken_Thompson_(sitting)_and_Dennis_Ritchie_at_PDP-11_(2876612463).jpg)

# Was sind Datenbanken?

## Zentraler Computer

- menschliche Anwesenheit erforderlich
- sorgfältig geordnet für effizienten Zugriff
  - Computer kann helfen, Ordnung zu erhalten



Bildquelle: [https://commons.wikimedia.org/wiki/File:Ken\\_Thompson\\_\(sitting\)\\_and\\_Dennis\\_Ritchie\\_at\\_PDP-11\\_\(2876612463\).jpg](https://commons.wikimedia.org/wiki/File:Ken_Thompson_(sitting)_and_Dennis_Ritchie_at_PDP-11_(2876612463).jpg)



# Was sind Datenbanken?

## Zentraler Computer

- menschliche Anwesenheit erforderlich
- sorgfältig geordnet für effizienten Zugriff
  - Computer kann helfen, Ordnung zu erhalten
- Jeweils 1 Person kann am Computer arbeiten.
  - Hardware-Unterstützung für Konsistenz

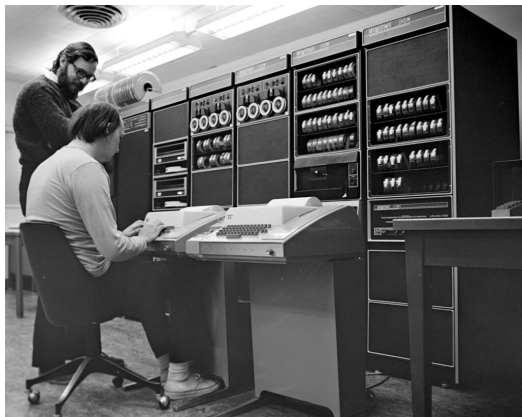


Bildquelle: [https://commons.wikimedia.org/wiki/File:Ken\\_Thompson\\_\(sitting\)\\_and\\_Dennis\\_Ritchie\\_at\\_PDP-11\\_\(2876612463\).jpg](https://commons.wikimedia.org/wiki/File:Ken_Thompson_(sitting)_and_Dennis_Ritchie_at_PDP-11_(2876612463).jpg)

# Was sind Datenbanken?

## Zentraler Computer

- menschliche Anwesenheit erforderlich
- sorgfältig geordnet für effizienten Zugriff  
→ Computer kann helfen, Ordnung zu erhalten
- Jeweils 1 Person kann am Computer arbeiten.  
→ Hardware-Unterstützung für Konsistenz
- Backups möglich



Bildquelle: [https://commons.wikimedia.org/wiki/File:Ken\\_Thompson\\_\(sitting\)\\_and\\_Dennis\\_Ritchie\\_at\\_PDP-11\\_\(2876612463\).jpg](https://commons.wikimedia.org/wiki/File:Ken_Thompson_(sitting)_and_Dennis_Ritchie_at_PDP-11_(2876612463).jpg)

# Was sind Datenbanken?

Zentraler Computer,  
Zugriff von Arbeitsplätzen aus



Bildquelle: [https://commons.wikimedia.org/wiki/File:Computergebouw\\_van\\_KLM\\_voor\\_automatische\\_boekingsmethode\\_Corda\\_in\\_Amstelveen.\\_,\\_Bestanddeelnr\\_923-3365.jpg](https://commons.wikimedia.org/wiki/File:Computergebouw_van_KLM_voor_automatische_boekingsmethode_Corda_in_Amstelveen._,_Bestanddeelnr_923-3365.jpg)

# Was sind Datenbanken?

Zentraler Computer,  
Zugriff von Arbeitsplätzen aus

- Arbeiten auf  
Entfernung möglich



Bildquelle: [https://commons.wikimedia.org/wiki/File:Computergebouw\\_van\\_KLM\\_voor\\_automatische\\_boekingsmethode\\_Corda\\_in\\_Amstelveen.\\_,\\_Bestanddeelnr\\_923-3365.jpg](https://commons.wikimedia.org/wiki/File:Computergebouw_van_KLM_voor_automatische_boekingsmethode_Corda_in_Amstelveen._,_Bestanddeelnr_923-3365.jpg)

# Was sind Datenbanken?

## Zentraler Computer, Zugriff von Arbeitsplätzen aus

- Arbeiten auf Entfernung möglich
- sorgfältig geordnet für effizienten Zugriff  
→ Computer kann helfen, Ordnung zu erhalten



# Was sind Datenbanken?

## Zentraler Computer, Zugriff von Arbeitsplätzen aus

- Arbeiten auf Entfernung möglich
- sorgfältig geordnet für effizienten Zugriff  
→ Computer kann helfen, Ordnung zu erhalten
- Mehrere Personen können unbemerkt gleichzeitig an denselben Daten arbeiten.  
→ Computer muß helfen, Konsistenz zu erhalten



# Was sind Datenbanken?

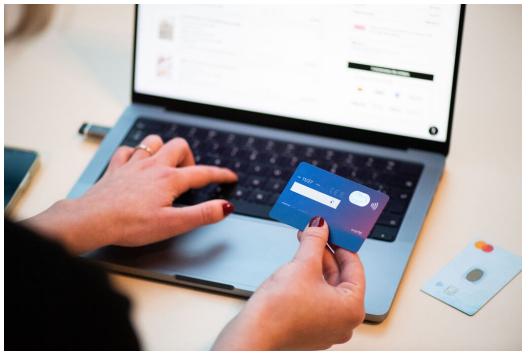
## Zentraler Computer, Zugriff von Arbeitsplätzen aus

- Arbeiten auf Entfernung möglich
- sorgfältig geordnet für effizienten Zugriff  
→ Computer kann helfen, Ordnung zu erhalten
- Mehrere Personen können unbemerkt gleichzeitig an denselben Daten arbeiten.  
→ Computer muß helfen, Konsistenz zu erhalten
- Backups möglich



# Was sind Datenbanken?

Zentraler Computer,  
öffentlicher Zugriff



Bildquelle: [https://commons.wikimedia.org/wiki/File:Shopping\\_online\\_with\\_bank\\_card.jpg](https://commons.wikimedia.org/wiki/File:Shopping_online_with_bank_card.jpg)



# Was sind Datenbanken?

## Zentraler Computer, öffentlicher Zugriff

- Selbstbedienung auf Entfernung möglich



Bildquelle: [https://commons.wikimedia.org/wiki/File:Shopping\\_online\\_with\\_bank\\_card.jpg](https://commons.wikimedia.org/wiki/File:Shopping_online_with_bank_card.jpg)

# Was sind Datenbanken?

## Zentraler Computer, öffentlicher Zugriff

- Selbstbedienung auf Entfernung möglich
- sorgfältig geordnet für effizienten Zugriff  
→ Computer muß selbständig Ordnung erhalten



Bildquelle: [https://commons.wikimedia.org/wiki/File:Shopping\\_online\\_with\\_bank\\_card.jpg](https://commons.wikimedia.org/wiki/File:Shopping_online_with_bank_card.jpg)

# Was sind Datenbanken?

## Zentraler Computer, öffentlicher Zugriff

- Selbstbedienung auf Entfernung möglich
- sorgfältig geordnet für effizienten Zugriff  
→ Computer muß selbständig Ordnung erhalten
- Mehrere Personen können unbemerkt gleichzeitig an denselben Daten arbeiten.  
→ Computer muß selbständig Konsistenz erhalten



# Was sind Datenbanken?

## Zentraler Computer, öffentlicher Zugriff

- Selbstbedienung auf Entfernung möglich
- sorgfältig geordnet für effizienten Zugriff  
→ Computer muß selbständig Ordnung erhalten
- Mehrere Personen können unbemerkt gleichzeitig an denselben Daten arbeiten.  
→ Computer muß selbständig Konsistenz erhalten
- Backups möglich



Bildquelle: [https://commons.wikimedia.org/wiki/File:Shopping\\_online\\_with\\_bank\\_card.jpg](https://commons.wikimedia.org/wiki/File:Shopping_online_with_bank_card.jpg)

# Was sind Datenbanken?

## Zentraler Computer, öffentlicher Zugriff

- Selbstbedienung auf Entfernung möglich
- sorgfältig geordnet für effizienten Zugriff  
→ Computer muß selbständig Ordnung erhalten
- Mehrere Personen können unbemerkt gleichzeitig an denselben Daten arbeiten.  
→ Computer muß selbständig Konsistenz erhalten
- Backups möglich
- Bessere Zugriffsmöglichkeiten → Datensicherheit wird Herausforderung



# Was ist Datensicherheit?

- Vertraulichkeit
- Integrität
- Verfügbarkeit

# Was ist Datensicherheit?

- Vertraulichkeit (CIA)
- Integrität (confidentiality)
- Verfügbarkeit (integrity)
- Verfügbarkheit (availability)

# Was ist Datensicherheit?

- Vertraulichkeit (CIA)
- Integrität (confidentiality) → Verschlüsselung (integrity)
- Verfügbarkeit (availability)



# Was ist Datensicherheit?

- |                   |                   |                       |
|-------------------|-------------------|-----------------------|
|                   | (CIA)             |                       |
| • Vertraulichkeit | (confidentiality) | → Verschlüsselung     |
| • Integrität      | (integrity)       | → Konsistenzprüfungen |
| • Verfügbarkeit   | (availability)    |                       |

# Was ist Datensicherheit?

- |                   |                   |                              |
|-------------------|-------------------|------------------------------|
|                   | (CIA)             |                              |
| • Vertraulichkeit | (confidentiality) | → Verschlüsselung            |
| • Integrität      | (integrity)       | → Konsistenzprüfungen        |
| • Verfügbarkeit   | (availability)    | → Backups, Ausfallsicherheit |

# Was ist Datensicherheit?

(CIA)

- Vertraulichkeit (confidentiality) → Verschlüsselung
- Integrität (integrity) → Konsistenzprüfungen
- Verfügbarkeit (availability) → Backups, Ausfallsicherheit
- Identifizierbarkeit  
(Authentizität, Nichtabstreitbarkeit, Zurechenbarkeit)  
→ Passwörter, Signaturen

# Was ist Datensicherheit?

(CIA)

- Vertraulichkeit (confidentiality) → Verschlüsselung
- Integrität (integrity) → Konsistenzprüfungen
- Verfügbarkeit (availability) → Backups, Ausfallsicherheit
  
- Identifizierbarkeit  
(Authentizität, Nichtabstreitbarkeit, Zurechenbarkeit)  
→ Passwörter, Signaturen  
bzw.
- Anonymität  
(plausible Abstreitbarkeit, Nichtzurechenbarkeit)  
→ Pseudonymisierung, Anonymisierung

# In dieser Lehrveranstaltung

- Kurzeinführung: Unix-Shell
- Kurzeinführung: TCP/IP
- relationale Datenbank-Management-Systeme (DBMS)
- SQL-Programmierung
- sonstige Datenbank-Management-Systeme
  
- Kurzeinführung: Kryptographie
- Kryptographie in der Praxis:  
Passwörter, Verschlüsselung, Signaturen,  
Schlüssel-Infrastrukturen
- Netzwerksicherheit
- Sicherheit von Web-Anwendungen
- Datenschutz

# In dieser Lehrveranstaltung

- Kurzeinführung: Unix-Shell
- Kurzeinführung: TCP/IP
- relationale Datenbank-Management-Systeme (DBMS)
- SQL-Programmierung
- sonstige Datenbank-Management-Systeme
  
- Kurzeinführung: Kryptographie
- Kryptographie in der Praxis:  
Passwörter, Verschlüsselung, Signaturen,  
Schlüssel-Infrastrukturen
- Netzwerksicherheit
- Sicherheit von Web-Anwendungen
- Datenschutz



Änderungen  
vorbehalten

# In dieser Lehrveranstaltung

## 3 Praktikumsversuche

1. Selbstbau eines einfachen DBMS
2. Selbstbau einer prototypischen, sicheren Datenbankanwendung
3. E-Mail-Verschlüsselung

**Prüfungsleistung:** Klausur



Änderungen  
vorbehalten

# In dieser Lehrveranstaltung

## 3 Praktikumsversuche

0. Praxiserfahrung mit Unix und TCP/IP
1. Selbstbau eines einfachen DBMS
2. Selbstbau einer prototypischen, sicheren Datenbankanwendung
3. E-Mail-Verschlüsselung

Keine festen Abgabetermine, sondern:  
Angebot zum betreuten Arbeiten im DV-Pool,  
Vorzeigen (Testieren) der Ergebnisse

**Prüfungsleistung:** Klausur



Änderungen  
vorbehalten



# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

- 1.1 Was sind Datenbanken?
- 1.2 Was ist Datensicherheit?
- 1.3 In dieser Lehrveranstaltung

## 2 Kurzeinführung Unix

- 2.1 Grundkonzepte
- 2.2 Die Kommandozeile: Grundlagen
- 2.3 Dateisysteme
- 2.4 Ein- und Ausgabeströme
- 2.5 Pipes
- 2.6 Verzweigungen und Schleifen

## 3 Kurzeinführung TCP/IP

...



Änderungen  
vorbehalten

## 2 Kurzeinführung Unix

### 2.1 Grundkonzepte

- 1965** Vorgänger: Multics (Multiplexed Information and Computing Service)  
„überladen“
- 1970** Unix: Einfachheit als Grundkonzept
- 1972** Umstellung auf neu entwickelte Programmiersprache C
- 1975** AT&T: Unix inkl. Quelltext für Universitäten
- 1977** Berkeley Software Distribution (BSD)
- 1983** GNU-Projekt
- 1987** Minix
- 1991** Linux
- 1993** FreeBSD, NetBSD
- 1994** OpenBSD
- 2000** Darwin (Mac OS X, BSD-basiert)
- 2008** Android (Linux-basiert)

## 2 Kurzeinführung Unix

### 2.1 Grundkonzepte

Unix und C: Einfachheit als Grundkonzept

- Vermeiden von Ausnahmen
- Baukastensystem

C: Hauptprogramm  
= „normale“ Funktion

```
int main (int argc, char **argv)
{
    printf ("Hello, _world!\n");
    return 0;
}
```

Unix: übergeordnetes Verzeichnis = „normales“ Verzeichnis

```
cassini/home/peter/foo> ls -la
insgesamt 24
drwxr-xr-x  2 peter peter  4096 Okt  6 13:30 .
drwxr-xr-x 172 peter peter 20480 Okt  6 13:30 ..
cassini/home/peter/foo> cd ..
cassini/home/peter>
```

## 2 Kurzeinführung Unix

### 2.1 Grundkonzepte

Unix und C: Einfachheit als Grundkonzept

- Vermeiden von Ausnahmen
- Baukastensystem

C: Bibliotheken

z. B.: `printf()` = „normale“ Funktion  
aus eine Bibliothek (`libc`)

Unix: Programme arbeiten zusammen

```
cassini/home/peter/bo> find . -name "*klausur*.tex" \  
| xargs grep -l "PBM-Datei"  
./2014ws/ainf/20150130.0/ainf-klausur-20150130.tex  
./2016ws/hp/20170920.0/klausur.tex  
./2016ws/hp/20170206.0/klausur.tex  
./2011ws/rarch/20120322.0/rarch-klausur-20120322.tex  
./2012ws/klausuren-gerwinski/rarch-klausur-20120322.tex  
./2013ws/ainf/20140918.0/ainf-klausur-20140918.tex  
./2017ws/hp/20180213.k1/klausur.tex  
./2017ws/hp/20180205/klausur.tex  
./2015ws/ainf/20160913/ainf-klausur-20160913.tex
```

## 2.2 Die Kommandozeile: Grundlagen

- Programm aufrufen: Namen eingeben, z. B.: `ls`
- Optionen: `ls -l`
- Lange Optionen (GNU-Konvention): `ls --help`
- Text schreiben: `echo "Hello, world!"`
- (String-)Variable setzen: `FOO=bar`
- Variable einlesen: `read FOO`
- Variable abrufen: `echo $FOO`
- Aus Sicherheitsgründen: `echo "$FOO"`

```
cassini/home/peter/bo> FOO=ls
cassini/home/peter/bo> echo "$FOO"
ls
cassini/home/peter/bo> $FOO
2011ws  2012ws  2013ws  doc          misc  projekte
2012ss  2013ss  briefe  material    orga
cassini/home/peter/bo>
```

## 2.2 Die Kommandozeile: Grundlagen

- Befehl zurückholen: Pfeiltasten  $\uparrow$ ,  $\downarrow$
- Befehl bearbeiten: Pfeiltasten  $\leftarrow$ ,  $\rightarrow$  usw.
- Befehl vervollständigen: TAB
- Befehl rückwärts suchen: Ctrl+R
- Bildschirm löschen: Ctrl+L
- Befehl abbrechen: Ctrl+C
  
- Hilfe-Option: `ls --help`
- Unix-Handbuch – *manual*: `man ls`  
(Beenden mit `q`)

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

9. Oktober 2024

# Vorab: Online-Werkzeuge

- Diese Veranstaltung findet **in Präsenz** statt.  
Wir versuchen aber, auch eine Online-Teilnahme zu ermöglichen.
- **Mumble**: Seminarraum 2  
Fragen: Mikrofon einschalten oder über den Chat  
Umfragen: über den Chat – **auch während der Präsenz-Veranstaltung**
- **VNC**: Kanal 6, Passwort: `testcvh`  
Eigenen Bildschirm freigeben: VNC-Software oder Web-Interface *yesVNC*  
Eigenes Kamerabild übertragen: Web-Interface *CVH-Camera*
- Allgemeine Informationen: <https://www.cvh-server.de/online-werkzeuge/>
- Notfall-Schnellzugang: <https://www.cvh-server.de/virtuelle-raeume/>  
Seminarraum 2, VNC-Passwort: `testcvh`
- **Lehrmaterialien**: <https://gitlab.cvh-server.de/pgerwinski/dbs>



# Was sind Datenbanken?

Datenbank = Datenbestand + System, um darauf zuzugreifen

Anforderungen an Datenbank [\[Wikipedia\]](#):

- effizient → Sortierung, Zugriffsalgorithmen
- widerspruchsfrei → automatische Konsistenzprüfungen
- dauerhaft → Backup, Ausfallsicherheit

# Was sind Datenbanken?

## Datenbank ohne Computer

- menschliche Anwesenheit erforderlich
- sorgfältig geordnet für effizienten Zugriff  
→ sorgfältiger Umgang erforderlich, um Ordnung zu erhalten
- Jeweils 1 Person kann an 1 Papier arbeiten.  
→ Hardware-Unterstützung für Konsistenz
- Kopien sehr aufwendig  
→ sorgfältige Archivierung erforderlich



Bildquelle: [https://commons.wikimedia.org/wiki/File:A\\_Day\\_in\\_the\\_Life\\_of\\_a\\_Wartime\\_Housewife-\\_Everyday\\_Life\\_in\\_London,\\_England,\\_1941\\_D2379.jpg](https://commons.wikimedia.org/wiki/File:A_Day_in_the_Life_of_a_Wartime_Housewife-_Everyday_Life_in_London,_England,_1941_D2379.jpg)

# Was sind Datenbanken?

## Zentraler Computer

- menschliche Anwesenheit erforderlich
- sorgfältig geordnet für effizienten Zugriff
  - Computer kann helfen, Ordnung zu erhalten
- Jeweils 1 Person kann am Computer arbeiten.
  - Hardware-Unterstützung für Konsistenz
- Backups möglich



Bildquelle: [https://commons.wikimedia.org/wiki/File:Ken\\_Thompson\\_\(sitting\)\\_and\\_Dennis\\_Ritchie\\_at\\_PDP-11\\_\(2876612463\).jpg](https://commons.wikimedia.org/wiki/File:Ken_Thompson_(sitting)_and_Dennis_Ritchie_at_PDP-11_(2876612463).jpg)

# Was sind Datenbanken?

## Zentraler Computer, Zugriff von Arbeitsplätzen aus

- Arbeiten auf Entfernung möglich
- sorgfältig geordnet für effizienten Zugriff  
→ Computer kann helfen, Ordnung zu erhalten
- Mehrere Personen können unbemerkt gleichzeitig an denselben Daten arbeiten.  
→ Computer muß helfen, Konsistenz zu erhalten
- Backups möglich



# Was sind Datenbanken?

## Zentraler Computer, öffentlicher Zugriff

- Selbstbedienung auf Entfernung möglich
- sorgfältig geordnet für effizienten Zugriff  
→ Computer muß selbständig Ordnung erhalten
- Mehrere Personen können unbemerkt gleichzeitig an denselben Daten arbeiten.  
→ Computer muß selbständig Konsistenz erhalten
- Backups möglich
- Bessere Zugriffsmöglichkeiten → Datensicherheit wird Herausforderung



Bildquelle: [https://commons.wikimedia.org/wiki/File:Shopping\\_online\\_with\\_bank\\_card.jpg](https://commons.wikimedia.org/wiki/File:Shopping_online_with_bank_card.jpg)

# Was ist Datensicherheit?

(CIA)

- Vertraulichkeit (confidentiality) → Verschlüsselung
- Integrität (integrity) → Konsistenzprüfungen
- Verfügbarkeit (availability) → Backups, Ausfallsicherheit

- Identifizierbarkeit  
(Authentizität, Nichtabstreitbarkeit, Zurechenbarkeit)  
→ Passwörter, Signaturen

bzw.

- Anonymität  
(plausible Abstreitbarkeit, Nichtzurechenbarkeit)  
→ Pseudonymisierung, Anonymisierung

# In dieser Lehrveranstaltung

- Kurzeinführung: Unix-Shell
- Kurzeinführung: TCP/IP
- relationale Datenbank-Management-Systeme (DBMS)
- SQL-Programmierung
- sonstige Datenbank-Management-Systeme
  
- Kurzeinführung: Kryptographie
- Kryptographie in der Praxis:  
Passwörter, Verschlüsselung, Signaturen,  
Schlüssel-Infrastrukturen
- Netzwerksicherheit
- Sicherheit von Web-Anwendungen
- Datenschutz



Änderungen  
vorbehalten

# In dieser Lehrveranstaltung

## 3 Praktikumsversuche

1. Selbstbau eines einfachen DBMS
2. Selbstbau einer prototypischen, sicheren Datenbankanwendung
3. E-Mail-Verschlüsselung

**Prüfungsleistung:** Klausur



Änderungen  
vorbehalten



# In dieser Lehrveranstaltung

## 3 Praktikumsversuche

0. Praxiserfahrung mit Unix und TCP/IP
1. Selbstbau eines einfachen DBMS
2. Selbstbau einer prototypischen, sicheren Datenbankanwendung
3. E-Mail-Verschlüsselung

Keine festen Abgabetermine, sondern:  
Angebot zum betreuten Arbeiten im DV-Pool,  
Vorzeigen (Testieren) der Ergebnisse

**Prüfungsleistung:** Klausur



Änderungen  
vorbehalten

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

- 1.1 Was sind Datenbanken?
- 1.2 Was ist Datensicherheit?
- 1.3 In dieser Lehrveranstaltung

## 2 Kurzeinführung Unix

- 2.1 Grundkonzepte
- 2.2 Die Kommandozeile: Grundlagen
- 2.3 Dateisysteme
- 2.4 Ein- und Ausgabeströme
- 2.5 Pipes
- 2.6 Verzweigungen und Schleifen

## 3 Kurzeinführung TCP/IP

...



Änderungen  
vorbehalten

## 2 Kurzeinführung Unix

### 2.1 Grundkonzepte

- 1965** Vorgänger: Multics (Multiplexed Information and Computing Service)  
„überladen“
- 1970** Unix: Einfachheit als Grundkonzept
- 1972** Umstellung auf neu entwickelte Programmiersprache C
- 1975** AT&T: Unix inkl. Quelltext für Universitäten
- 1977** Berkeley Software Distribution (BSD)
- 1983** GNU-Projekt
- 1987** Minix
- 1991** Linux
- 1993** FreeBSD, NetBSD
- 1994** OpenBSD
- 2000** Darwin (Mac OS X, BSD-basiert)
- 2008** Android (Linux-basiert)

## 2 Kurzeinführung Unix

### 2.1 Grundkonzepte

Unix und C: Einfachheit als Grundkonzept

- Vermeiden von Ausnahmen
- Baukastensystem

C: Hauptprogramm  
= „normale“ Funktion

```
int main (int argc, char **argv)
{
    printf ("Hello, _world!\n");
    return 0;
}
```

Unix: übergeordnetes Verzeichnis = „normales“ Verzeichnis

```
cassini/home/peter/foo> ls -la
insgesamt 24
drwxr-xr-x  2 peter peter  4096 Okt  6 13:30 .
drwxr-xr-x 172 peter peter 20480 Okt  6 13:30 ..
cassini/home/peter/foo> cd ..
cassini/home/peter>
```

## 2 Kurzeinführung Unix

### 2.1 Grundkonzepte

Unix und C: Einfachheit als Grundkonzept

- Vermeiden von Ausnahmen
- Baukastensystem

C: Bibliotheken

z. B.: `printf()` = „normale“ Funktion  
aus eine Bibliothek (`libc`)

Unix: Programme arbeiten zusammen

```
cassini/home/peter/bo> find . -name "*klausur*.tex" \  
| xargs grep -l "PBM-Datei"  
./2014ws/ainf/20150130.0/ainf-klausur-20150130.tex  
./2016ws/hp/20170920.0/klausur.tex  
./2016ws/hp/20170206.0/klausur.tex  
./2011ws/rarch/20120322.0/rarch-klausur-20120322.tex  
./2012ws/klausuren-gerwinski/rarch-klausur-20120322.tex  
./2013ws/ainf/20140918.0/ainf-klausur-20140918.tex  
./2017ws/hp/20180213.k1/klausur.tex  
./2017ws/hp/20180205/klausur.tex  
./2015ws/ainf/20160913/ainf-klausur-20160913.tex
```

## 2.2 Die Kommandozeile: Grundlagen

- Programm aufrufen: Namen eingeben, z. B.: `ls`
- Optionen: `ls -l`
- Lange Optionen (GNU-Konvention): `ls --help`
- Text schreiben: `echo "Hello, world!"`
- (String-)Variable setzen: `FOO=bar`
- Variable einlesen: `read FOO`
- Variable abrufen: `echo $FOO`
- Aus Sicherheitsgründen: `echo "$FOO"`

```
cassini/home/peter/bo> FOO=ls
cassini/home/peter/bo> echo "$FOO"
ls
cassini/home/peter/bo> $FOO
2011ws  2012ws  2013ws  doc          misc  projekte
2012ss  2013ss  briefe  material    orga
cassini/home/peter/bo>
```

## 2.2 Die Kommandozeile: Grundlagen

- Befehl zurückholen: Pfeiltasten  $\uparrow$ ,  $\downarrow$
- Befehl bearbeiten: Pfeiltasten  $\leftarrow$ ,  $\rightarrow$  usw.
- Befehl vervollständigen: TAB
- Befehl rückwärts suchen: Ctrl+R
- Bildschirm löschen: Ctrl+L
- Befehl abbrechen: Ctrl+C
  
- Hilfe-Option: `ls --help`
- Unix-Handbuch – *manual*: `man ls`  
(Beenden mit `q`)

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

- 1.1 Was sind Datenbanken?
- 1.2 Was ist Datensicherheit?
- 1.3 In dieser Lehrveranstaltung

## 2 Kurzeinführung Unix

- 2.1 Grundkonzepte
- 2.2 Die Kommandozeile: Grundlagen
- 2.3 Dateisysteme
- 2.4 Ein- und Ausgabeströme
- 2.5 Pipes
- 2.6 Verzweigungen und Schleifen

## 3 Kurzeinführung TCP/IP

...



Änderungen  
vorbehalten



## 2.2 Die Kommandozeile: Grundlagen

- Verzeichnisse für Programme: `echo $PATH`
- Programm in explizitem Verzeichnis aufrufen: `/bin/ls -l`
- Programm im aktuellen Verzeichnis aufrufen: `./hello`

**MS-DOS:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen  
*oder* sich im aktuellen Verzeichnis befinden.

**Unix:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen.

—→ Vermeiden von Ausnahmen

Das aktuelle Verzeichnis ( `.` ) *kann* im `PATH` stehen,  
muß dies aber nicht.

## 2.2 Die Kommandozeile: Grundlagen

- Verzeichnisse für Programme: `echo $PATH`
- Programm in explizitem Verzeichnis aufrufen: `/bin/ls -l`
- Programm im aktuellen Verzeichnis aufrufen: `./hello`

**MS-DOS:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen  
*oder* sich im aktuellen Verzeichnis befinden.

**Unix:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen.

—→ Vermeiden von Ausnahmen

Das aktuelle Verzeichnis ( `.` ) *kann* im `PATH` stehen,  
muß dies aber nicht –  
und sollte es aus Sicherheitsgründen auch nicht.

## 2.3 Dateisysteme

- Dateien listen: `ls`  
langes Listenformat: `ls -l`  
rückwärts nach Zeit sortiert: `ls -lrt`
- Datei ausgeben: `cat hello.c`
- Datei anzeigen: `less hello.c`

## 2.3 Dateisysteme

- Arbeitsverzeichnis anzeigen: `pwd`
- Arbeitsverzeichnis wechseln: `cd script`  
(*kein* Programm, sondern Shell-Befehl)
- übergeordnetes Verzeichnis: `cd ..`
- eigenes *Home*-Verzeichnis: `cd`
- Wurzelverzeichnis: `cd /`
- wieder zurück: `cd -`

```
cassini/home/peter/bo/2013ss/net/script> cd /usr/bin
cassini/usr/bin> cd ../lib
cassini/usr/lib> cd
cassini/home/peter>
```

## 2.3 Dateisysteme

- Dateien kopieren (*copy*): `cp`
- Dateien verschieben/umbenennen (*move*): `mv`
- Dateien löschen (*remove*): `rm`

```
cassini/home/peter> cp -p foo/test.txt
cp: missing destination file operand after `foo/test.txt'
Try `cp --help' for more information.
cassini/home/peter> cp -p foo/test.txt .
cassini/home/peter> mv test.txt bla.txt
cassini/home/peter> cat bla.txt
Dies ist ein Test.
cassini/home/peter> rm bla.txt
cassini/home/peter>
```

Aktuelles Verzeichnis: `.`

## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

## 2.3 Dateisysteme

- Zugriffsrechte

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



Benutzer (u – *user*) darf lesen und schreiben


## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



Gruppe (g – *group*) darf lesen



## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



alle anderen (o – *other*) dürfen lesen

## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
...
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

- Zugriffsrechte ändern:

```
chmod o-r es-20191009.tex – Lesezugriff entziehen
chmod g+w es-20191009.tex – Schreibzugriff gewähren
chmod 640 es-20191009.tex – auf -rw-r----- setzen
```

## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
...
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

- Zugriffsrechte ändern:

```
chmod o-r es-20191009.tex – Lesezugriff entziehen
chmod g+w es-20191009.tex – Schreibzugriff gewähren
chmod 640 es-20191009.tex – auf -rw-r----- setzen
                        6   4   0
```

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

16. Oktober 2024

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

- 1.1 Was sind Datenbanken?
- 1.2 Was ist Datensicherheit?
- 1.3 In dieser Lehrveranstaltung

## 2 Kurzeinführung Unix

- 2.1 Grundkonzepte
- 2.2 Die Kommandozeile: Grundlagen
- 2.3 Dateisysteme
- 2.4 Ein- und Ausgabeströme
- 2.5 Pipes
- 2.6 Verzweigungen und Schleifen

## 3 Kurzeinführung TCP/IP

...



Änderungen  
vorbehalten

# 1 Einführung

## 1.3 In dieser Lehrveranstaltung

### 3 Praktikumsversuche

0. Praxiserfahrung mit Unix und TCP/IP
1. Selbstbau eines einfachen DBMS
2. Selbstbau einer prototypischen, sicheren Datenbankanwendung
3. E-Mail-Verschlüsselung

Keine festen Abgabetermine, sondern:  
Angebot zum betreuten Arbeiten im DV-Pool,  
Vorzeigen (Testieren) der Ergebnisse

**Prüfungsleistung:** Klausur



Änderungen  
vorbehalten

## 2.2 Die Kommandozeile: Grundlagen

- Verzeichnisse für Programme: `echo $PATH`
- Programm in explizitem Verzeichnis aufrufen: `/bin/ls -l`
- Programm im aktuellen Verzeichnis aufrufen: `./hello`

**MS-DOS:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen  
oder sich im aktuellen Verzeichnis befinden.

**Unix:** Ausführbare Programme werden gefunden,  
wenn sie im `PATH` stehen.

→ Vermeiden von Ausnahmen

Das aktuelle Verzeichnis ( `.` ) *kann* im `PATH` stehen,  
muß dies aber nicht –  
und sollte es aus Sicherheitsgründen auch nicht.

## 2.3 Dateisysteme

- Dateien listen: `ls`  
langes Listenformat: `ls -l`  
rückwärts nach Zeit sortiert: `ls -lrt`
- Datei ausgeben: `cat hello.c`
- Datei anzeigen: `less hello.c`



## 2.3 Dateisysteme

- Arbeitsverzeichnis anzeigen: `pwd`
- Arbeitsverzeichnis wechseln: `cd script`  
(*kein* Programm, sondern Shell-Befehl)
- übergeordnetes Verzeichnis: `cd ..`
- eigenes *Home*-Verzeichnis: `cd`
- Wurzelverzeichnis: `cd /`
- wieder zurück: `cd -`

```
cassini/home/peter/bo/2013ss/net/script> cd /usr/bin
cassini/usr/bin> cd ../lib
cassini/usr/lib> cd
cassini/home/peter>
```

## 2.3 Dateisysteme

- Dateien kopieren (*copy*): `cp`
- Dateien verschieben/umbenennen (*move*): `mv`
- Dateien löschen (*remove*): `rm`

```
cassini/home/peter> cp -p foo/test.txt
cp: missing destination file operand after `foo/test.txt'
Try `cp --help' for more information.
cassini/home/peter> cp -p foo/test.txt .
cassini/home/peter> mv test.txt bla.txt
cassini/home/peter> cat bla.txt
Dies ist ein Test.
cassini/home/peter> rm bla.txt
cassini/home/peter>
```

Aktuelles Verzeichnis: `.`

## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

## 2.3 Dateisysteme

- Zugriffsrechte

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



Benutzer (u – *user*) darf lesen und schreiben


## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



Gruppe (g – *group*) darf lesen

## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



alle anderen (o – *other*) dürfen lesen

## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
...
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

- Zugriffsrechte ändern:

```
chmod o-r es-20191009.tex – Lesezugriff entziehen
chmod g+w es-20191009.tex – Schreibzugriff gewähren
chmod 640 es-20191009.tex – auf -rw-r----- setzen
```

## 2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
...
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

- Zugriffsrechte ändern:

```
chmod o-r es-20191009.tex – Lesezugriff entziehen
chmod g+w es-20191009.tex – Schreibzugriff gewähren
chmod 640 es-20191009.tex – auf -rw-r----- setzen
                        6   4   0
```



## 2.3 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2019ws/es/20191002> cat test2.txt
```

```
ls -l
```

```
cassini/home/peter/bo/2019ws/es/20191002> chmod +x test2.txt
```

```
cassini/home/peter/bo/2019ws/es/20191002> ls -l test2.txt
```

```
-rwxr-xr-x 1 peter peter 6 Okt  2 13:43 test2.txt
```

```
cassini/home/peter/bo/2019ws/es/20191002> ./test2.txt
```

```
insgesamt 4828
```

```
lrwxrwxrwx 1 peter peter      18 Apr 13  2016 csa2.jpg -> ..
```

```
-rw-r--r-- 1 peter peter 4619138 Okt  8 21:28 es-20191002.pdf
```

```
...
```

- ausführbare Textdateien: *Skripte*

hier: ausführbare Textdatei mit Shell-Befehlen  
(ohne spezielle Kennung): Shell-Skript

## 2.3 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2019ws/es/20191002> cat test2.txt
ls -l
cassini/home/peter/bo/2019ws/es/20191002> chmod +x test2.txt
cassini/home/peter/bo/2019ws/es/20191002> ls -l test2.txt
-rwxr-xr-x 1 peter peter 6 Okt  2 13:43 test2.txt
cassini/home/peter/bo/2019ws/es/20191002> ./test2.txt
insgesamt 4828
lrwxrwxrwx 1 peter peter      18 Apr 13  2016 csa2.jpg -> ..
-rw-r--r-- 1 peter peter 4619138 Okt  8 21:28 es-20191002.pdf
...
```

- ausführbare Textdateien: *Skripte*

hier: ausführbare Textdatei mit Shell-Befehlen  
(ohne spezielle Kennung): Shell-Skript

Kennung: 1. Zeile enthält `#!` und den Interpreter,  
z. B. `#!/bin/bash`

## 2.3 Dateisysteme

- Datenträger in Verzeichnis *einhängen*: `mount`

```
cassini/home/peter> ls /media/usb1/  
cassini/home/peter> mount /media/usb1  
cassini/home/peter> ls /media/usb1/  
es-20191002.pdf  hello.c  hexapode  KIS-Bericht.pdf  
cassini/home/peter> umount /media/usb1  
cassini/home/peter> ls /media/usb1/  
cassini/home/peter>
```

## 2.3 Dateisysteme

- *Symbolische Verknüpfungen – symbolic links*

Verweis auf die eigentliche Datei

→ Wenn man die Datei löscht, zeigt der Link ins Leere.

Verknüpfung anlegen: `ln -s datei link`

(Richtung: wie bei `cp`)

Beispiel: `ln -s ../common/GNU-GPL-3 gpl.txt`

## 2.3 Dateisysteme

- *Symbolische Verknüpfungen – symbolic links*

Verweis auf die eigentliche Datei

→ Wenn man die Datei löscht, zeigt der Link ins Leere.

Verknüpfung anlegen: `ln -s datei link`

(Richtung: wie bei `cp`)

Beispiel: `ln -s ../common/GNU-GPL-3 gpl.txt`

- *Harte Verknüpfungen – hard links*

Dieselben Daten auf dem Datenträger

sind unter mehreren Namen verfügbar.

→ Wenn man einen löscht, sind die Daten noch da.

```
cassini/home/peter/bo/2019ws/es/20191002> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter      1202 Okt  2 13:35 shell-06.tx
```

```
drwxr-xr-x 2 peter peter     4096 Okt  2 13:16 test
```



Anzahl der („harten“) Links auf diese Datei / dieses Verzeichnis

## 2.3 Dateisysteme

- `grep`: Dateien durchsuchen

```
cassini/home/peter/bo/2019ws/es/20191002> grep gcc *.txt  
shell-03.txt: cassini/...> gcc -Wall -O hello.c -o hello
```

## 2.3 Dateisysteme

- **find**: Dateien anhand ihrer Eigenschaften suchen

```
$ find . -name "*.txt"
./shell-06.txt
./shell-03.txt
./shell-05.txt
./test.txt
./test/test.txt
...
$ find . -name "*.txt" -perm /u+x
./test2.txt
$ find . -name "*.txt" -perm /u+x -exec ls -l {} \;
-rwxr-xr-x 1 peter peter 6 Okt  2 13:43 ./test2.txt
```

## 2.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```



## 2.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

- Standard-Ausgabe an Datei anhängen

```
$ echo "Dies ist noch ein Test." >> test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

```
Dies ist noch ein Test.
```

## 2.4 Ein- und Ausgabeströme

- Fehler-Ausgabe in Datei umleiten

```
$ cat gibtsnicht.txt > fehler.txt
cat: gibtsnicht.txt: No such file or directory
$ cat fehler.txt
$ cat gibtsnicht.txt 2> fehler.txt
$ cat fehler.txt
cat: gibtsnicht.txt: No such file or directory
```

## 2.4 Ein- und Ausgabeströme

- Standard-Eingabe aus Datei lesen

```
$ bc
bc 1.06.95
Copyright [...] 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
2 + 2
4
$ echo "2 + 2" > test.bc
$ bc < test.bc
4
```

## 2.5 Pipes

Standard-Ausgabe von Programm A  
wird zu Standard-Eingabe von Programm B

```
$ echo "2 + 2" | bc  
4
```

—→ sehr mächtiger „Baukasten“

## 2.5 Pipes

- `sed`: *stream editor*

Suchen und Ersetzen (und noch viel mehr)

```
$ echo "Schlimmer geht nimmer." | sed -e 's/nim/im/g'  
Schlimmer geht immer.
```

## 2.5 Pipes

- **grep**: Standard-Eingabe durchsuchen

```
$ ls | grep slides
```

```
pgslides.sty
```

```
$ ls *.pdf | grep -v logo
```

```
es-20191002.pdf
```

```
Zeichen_123.pdf
```

```
$ ls -l $(ls *.pdf | grep -v logo)
```

```
-rw-r--r-- 1 ... 4619138 Okt 8 21:28 es-20191002.pdf
```

```
lrwxrwxrwx 1 ...          25 Okt 3  2016 Zeichen_123.pdf -> ...
```

## 2.6 Verzweigungen und Schleifen

```
$ if grep Blubb test.txt; then echo "gefunden"; \  
    else echo "nicht gefunden"; fi  
nicht gefunden  
$ for x in foo bar baz; do echo $x; done  
foo  
bar  
baz
```

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

- 1.1 Was sind Datenbanken?
- 1.2 Was ist Datensicherheit?
- 1.3 In dieser Lehrveranstaltung

## 2 Kurzeinführung Unix

- 2.1 Grundkonzepte
- 2.2 Die Kommandozeile: Grundlagen
- 2.3 Dateisysteme
- 2.4 Ein- und Ausgabeströme
- 2.5 Pipes
- 2.6 Verzweigungen und Schleifen

## 3 Kurzeinführung TCP/IP

...



Änderungen  
vorbehalten



# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

30. Oktober 2024

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

## 2 Kurzeinführung Unix

2.1 Grundkonzepte

2.2 Die Kommandozeile: Grundlagen

2.3 Dateisysteme

2.4 Ein- und Ausgabeströme

2.5 Pipes

2.6 Verzweigungen und Schleifen

## 3 Kurzeinführung TCP/IP

3.1 MAC-Adressen

3.2 IP-Adressen

3.3 TCP- und UDP-Ports

...

...



Änderungen  
vorbehalten

## 2.4 Dateisysteme

- Dateien listen: `ls`  
langes Listenformat: `ls -l`  
rückwärts nach Zeit sortiert: `ls -lrt`
- Datei ausgeben: `cat hello.c`
- Datei anzeigen: `less hello.c`

## 2.4 Dateisysteme

- Arbeitsverzeichnis anzeigen: `pwd`
- Arbeitsverzeichnis wechseln: `cd script`  
(*kein* Programm, sondern Shell-Befehl)
- übergeordnetes Verzeichnis: `cd ..`
- eigenes *Home*-Verzeichnis: `cd`
- Wurzelverzeichnis: `cd /`
- wieder zurück: `cd -`

```
cassini/home/peter/bo/2013ss/net/script> cd /usr/bin
cassini/usr/bin> cd ../lib
cassini/usr/lib> cd
cassini/home/peter>
```

## 2.4 Dateisysteme

- Dateien kopieren (*copy*): `cp`
- Dateien verschieben/umbenennen (*move*): `mv`
- Dateien löschen (*remove*): `rm`

```
cassini/home/peter> cp -p foo/test.txt
cp: missing destination file operand after `foo/test.txt'
Try `cp --help' for more information.
cassini/home/peter> cp -p foo/test.txt .
cassini/home/peter> mv test.txt bla.txt
cassini/home/peter> cat bla.txt
Dies ist ein Test.
cassini/home/peter> rm bla.txt
cassini/home/peter>
```

Aktuelles Verzeichnis: `.`

## 2.4 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

## 2.4 Dateisysteme

- Zugriffsrechte

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



Benutzer (u – *user*) darf lesen und schreiben


## 2.4 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



Gruppe (g – *group*) darf lesen



## 2.4 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



alle anderen (o – *other*) dürfen lesen

## 2.4 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
...
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

- Zugriffsrechte ändern:

```
chmod o-r es-20191009.tex – Lesezugriff entziehen
chmod g+w es-20191009.tex – Schreibzugriff gewähren
chmod 640 es-20191009.tex – auf rw-r---- setzen
                        6   4   0
```

## 2.4 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2019ws/es/20191002> cat test2.txt
ls -l
cassini/home/peter/bo/2019ws/es/20191002> chmod +x test2.txt
cassini/home/peter/bo/2019ws/es/20191002> ls -l test2.txt
-rwxr-xr-x 1 peter peter 6 Okt  2 13:43 test2.txt
cassini/home/peter/bo/2019ws/es/20191002> ./test2.txt
insgesamt 4828
lrwxrwxrwx 1 peter peter      18 Apr 13  2016 csa2.jpg -> ..
-rw-r--r-- 1 peter peter 4619138 Okt  8 21:28 es-20191002.pdf
...
```

- ausführbare Textdateien: *Skripte*

hier: ausführbare Textdatei mit Shell-Befehlen  
(ohne spezielle Kennung): Shell-Skript

Kennung: 1. Zeile enthält `#!` und den Interpreter,  
z. B. `#!/bin/bash`

## 2.4 Dateisysteme

- Datenträger in Verzeichnis *einhängen*: `mount`

```
cassini/home/peter> ls /media/usb1/  
cassini/home/peter> mount /media/usb1  
cassini/home/peter> ls /media/usb1/  
es-20191002.pdf  hello.c  hexapode  KIS-Bericht.pdf  
cassini/home/peter> umount /media/usb1  
cassini/home/peter> ls /media/usb1/  
cassini/home/peter>
```

## 2.4 Dateisysteme

- *Symbolische Verknüpfungen – symbolic links*

Verweis auf die eigentliche Datei

→ Wenn man die Datei löscht, zeigt der Link ins Leere.

Verknüpfung anlegen: `ln -s datei link`

(Richtung: wie bei `cp`)

Beispiel: `ln -s ../common/GNU-GPL-3 gpl.txt`

- *Harte Verknüpfungen – hard links*

Dieselben Daten auf dem Datenträger

sind unter mehreren Namen verfügbar.

→ Wenn man einen löscht, sind die Daten noch da.

```
cassini/home/peter/bo/2019ws/es/20191002> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter    1202 Okt  2 13:35 shell-06.tx
```

```
drwxr-xr-x 2 peter peter   4096 Okt  2 13:16 test
```



Anzahl der („harten“) Links auf diese Datei / dieses Verzeichnis

## 2.4 Dateisysteme

- `grep`: Dateien durchsuchen

```
cassini/home/peter/bo/2019ws/es/20191002> grep gcc *.txt  
shell-03.txt: cassini/...> gcc -Wall -O hello.c -o hello
```

## 2.4 Dateisysteme

- **find**: Dateien anhand ihrer Eigenschaften suchen

```
$ find . -name "*.txt"
./shell-06.txt
./shell-03.txt
./shell-05.txt
./test.txt
./test/test.txt
...
$ find . -name "*.txt" -perm /u+x
./test2.txt
$ find . -name "*.txt" -perm /u+x -exec ls -l {} \;
-rwxr-xr-x 1 peter peter 6 Okt  2 13:43 ./test2.txt
```

## 2.5 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```



## 2.5 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

- Standard-Ausgabe an Datei anhängen

```
$ echo "Dies ist noch ein Test." >> test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

```
Dies ist noch ein Test.
```

## 2.5 Ein- und Ausgabeströme

- Fehler-Ausgabe in Datei umleiten

```
$ cat gibtsnicht.txt > fehler.txt
cat: gibtsnicht.txt: No such file or directory
$ cat fehler.txt
$ cat gibtsnicht.txt 2> fehler.txt
$ cat fehler.txt
cat: gibtsnicht.txt: No such file or directory
```

## 2.5 Ein- und Ausgabeströme

- Standard-Eingabe aus Datei lesen

```
$ bc
bc 1.06.95
Copyright [...] 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
2 + 2
4
$ echo "2 + 2" > test.bc
$ bc < test.bc
4
```

## 2.5 Ein- und Ausgabeströme

- Standard-Ausgabe in Befehlszeile übernehmen

```
$ find . -name "dbs-2024*.tex"  
./20241002/dbs-2024ws-p0.tex  
./20241002/dbs-20241002.tex  
./20241030/dbs-20241030.tex  
./20241016/dbs-20241016.tex  
./20241009/dbs-20241009.tex  
$ grep -l "Sägefisch" $(find . -name "dbs-2024*.tex")  
./20241030/dbs-20241030.tex
```

Alternative Schreibweise: `...` statt `$(...)`

## 2.6 Pipes

Standard-Ausgabe von Programm A  
wird zu Standard-Eingabe von Programm B

```
$ echo "2 + 2" | bc  
4
```

—→ sehr mächtiger „Baukasten“

## 2.6 Pipes

- `sed`: *stream editor*

Suchen und Ersetzen (und noch viel mehr)

```
$ echo "Schlimmer geht nimmer." | sed -e 's/nim/im/g'  
Schlimmer geht immer.
```

## 2.6 Pipes

- **grep**: Standard-Eingabe durchsuchen

```
$ ls | grep slides
```

```
pgslides.sty
```

```
$ ls *.pdf | grep -v logo
```

```
es-20191002.pdf
```

```
Zeichen_123.pdf
```

```
$ ls -l $(ls *.pdf | grep -v logo)
```

```
-rw-r--r-- 1 ... 4619138 Okt 8 21:28 es-20191002.pdf
```

```
lrwxrwxrwx 1 ...          25 Okt 3  2016 Zeichen_123.pdf -> ...
```

## 2.7 Verzweigungen und Schleifen

```
$ if grep Blubb test.txt; then echo "gefunden"; \  
    else echo "nicht gefunden"; fi  
nicht gefunden  
$ for x in foo bar baz; do echo $x; done  
foo  
bar  
baz
```



# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

## 2 Kurzeinführung Unix

2.1 Grundkonzepte

2.2 Die Kommandozeile: Grundlagen

2.3 Dateisysteme

2.4 Ein- und Ausgabeströme

2.5 Pipes

2.6 Verzweigungen und Schleifen

## 3 Kurzeinführung TCP/IP

3.1 MAC-Adressen

3.2 IP-Adressen

3.3 TCP- und UDP-Ports

...

...



Änderungen  
vorbehalten

## 3.1 MAC-Adressen

MAC = Media Access Control

MAC-Adresse = Hardware-Adresse = Ethernet-Adresse

- `ip neigh`  
`arp`

## 3.2 IP-Adressen

- `ip addr` (Linux)  
  `ifconfig` (Unix allgemein)  
  `ipconfig` (MS Windows)
- `ip addr add <Netz>`
- `ip link`
- `ping <IP-Adresse>`

```
# ifconfig
```

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    [...]
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.42.101  netmask 255.255.255.0
                                broadcast 192.168.42.255
    ether be:3f:ca:aa:7e:51 txqueuelen 1000  (Ethernet)
    [...]
```

## 3.2 IP-Adressen

IPv4-Adressen:

- 32 Bit
- dezimal, 4 Gruppen zu je 8 Bit (0–255), durch Punkte getrennt

IPv6-Adressen:

- 128 Bit
- hexadezimal, 8 Gruppen zu je 4 Hex-Ziffern, durch Doppelpunkte getrennt
- Führende Nullen dürfen weggelassen werden.
- Zwei Doppelpunkte bedeuten: Mit Nullen auffüllen.
- Literatur und Beispiel: <https://de.wikipedia.org/wiki/IPv6>

## 3.3 TCP- und UDP-Ports

- `nc <IP> <Port>`  
Verbindung zu Programm <Port> auf Rechner <IP> aufnehmen
- `nc -l <Port>` oder `nc -p <Port> -l`  
auf eingehende Verbindungen warten („lauschen“)
- `nc -c <Programm>`  
Programm statt Mensch kommunizieren lassen
- TCP-Ports: Verbindungskonzept, Netzwerk prüft
- UDP-Ports: einzelne Pakete, Anwendung muß selbst prüfen
- ICMP: keine Ports, nur Rechner:  
Erreichbarkeit, Eigenschaften der Übertragung

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

6. November 2024

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

## 2 Kurzeinführung Unix

2.1 Grundkonzepte

2.2 Die Kommandozeile: Grundlagen

2.3 Dateisysteme

2.4 Ein- und Ausgabeströme

2.5 Pipes

2.6 Verzweigungen und Schleifen

## 3 Kurzeinführung TCP/IP

3.1 MAC-Adressen

3.2 IP-Adressen

3.3 TCP- und UDP-Ports

...

...



Änderungen  
vorbehalten

## 2.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

- Standard-Ausgabe an Datei anhängen

```
$ echo "Dies ist noch ein Test." >> test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

```
Dies ist noch ein Test.
```



## 2.4 Ein- und Ausgabeströme

- Fehler-Ausgabe in Datei umleiten

```
$ cat gibtsnicht.txt > fehler.txt
cat: gibtsnicht.txt: No such file or directory
$ cat fehler.txt
$ cat gibtsnicht.txt 2> fehler.txt
$ cat fehler.txt
cat: gibtsnicht.txt: No such file or directory
```

## 2.4 Ein- und Ausgabeströme

- Standard-Eingabe aus Datei lesen

```
$ bc
bc 1.06.95
Copyright [...] 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
2 + 2
4
$ echo "2 + 2" > test.bc
$ bc < test.bc
4
```

## 2.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Befehlszeile übernehmen

```
$ find . -name "dbs-2024*.tex"
./20241002/dbs-2024ws-p0.tex
./20241002/dbs-20241002.tex
./20241030/dbs-20241030.tex
./20241016/dbs-20241016.tex
./20241009/dbs-20241009.tex
./20241106/dbs-20241106.tex
$ grep -l "Sägefisch" $(find . -name "dbs-2024*.tex")
./20241030/dbs-20241030.tex
./20241030/dbs-20241106.tex
```

Alternative Schreibweise: ``...`` statt `$(...)`

## 2.5 Pipes

Standard-Ausgabe von Programm A  
wird zu Standard-Eingabe von Programm B

```
$ echo "2 + 2" | bc  
4
```

—→ sehr mächtiger „Baukasten“

## 2.5 Pipes

- `sed`: *stream editor*

Suchen und Ersetzen (und noch viel mehr)

```
$ echo "Schlimmer geht nimmer." | sed -e 's/nim/im/g'  
Schlimmer geht immer.
```

## 2.5 Pipes

- **grep**: Standard-Eingabe durchsuchen

```
$ ls | grep slides
```

```
pgslides.sty
```

```
$ ls *.pdf | grep -v logo
```

```
es-20191002.pdf
```

```
Zeichen_123.pdf
```

```
$ ls -l $(ls *.pdf | grep -v logo)
```

```
-rw-r--r-- 1 ... 4619138 Okt 8 21:28 es-20191002.pdf
```

```
lrwxrwxrwx 1 ...          25 Okt 3  2016 Zeichen_123.pdf -> ...
```

## 2.6 Verzweigungen und Schleifen

```
$ if grep Blubb test.txt; then echo "gefunden"; \  
    else echo "nicht gefunden"; fi  
nicht gefunden  
$ for x in foo bar baz; do echo $x; done  
foo  
bar  
baz
```

## 3 Kurzeinführung TCP/IP

### 3.1 MAC-Adressen

MAC = Media Access Control

MAC-Adresse = Hardware-Adresse = Ethernet-Adresse

- `ip neigh`  
`arp`



## 3.2 IP-Adressen

- `ip addr` (Linux)  
  `ifconfig` (Unix allgemein)  
  `ipconfig` (MS Windows)
- `ip addr add <Netz>`
- `ip link`
- `ping <IP-Adresse>`

```
# ifconfig
```

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    [...]
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.42.101  netmask 255.255.255.0
                                broadcast 192.168.42.255
    ether be:3f:ca:aa:7e:51 txqueuelen 1000  (Ethernet)
    [...]
```

## 3.2 IP-Adressen

IPv4-Adressen:

- 32 Bit
- dezimal, 4 Gruppen zu je 8 Bit (0–255), durch Punkte getrennt

IPv6-Adressen:

- 128 Bit
- hexadezimal, 8 Gruppen zu je 4 Hex-Ziffern, durch Doppelpunkte getrennt
- Führende Nullen dürfen weggelassen werden.
- Zwei Doppelpunkte bedeuten: Mit Nullen auffüllen.
- Literatur und Beispiel: <https://de.wikipedia.org/wiki/IPv6>

## 3.3 TCP- und UDP-Ports

- `nc <IP> <Port>`  
Verbindung zu Programm <Port> auf Rechner <IP> aufnehmen
- `nc -l <Port>` oder `nc -p <Port> -l`  
auf eingehende Verbindungen warten („lauschen“)
- `nc -c <Programm>`  
Programm statt Mensch kommunizieren lassen
- TCP-Ports: Verbindungskonzept, Netzwerk prüft
- UDP-Ports: einzelne Pakete, Anwendung muß selbst prüfen
- ICMP: keine Ports, nur Rechner:  
Erreichbarkeit, Eigenschaften der Übertragung

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse

## 3.4 TCP-Protokolle

- **HTTP**

GET / HTTP/1.1

Host: www.hs-bochum.de

(Leerzeile)

## 3.4 TCP-Protokolle

- **HTTP**

```
GET / HTTP/1.1
```

```
Host: www.hs-bochum.de
```

```
(Leerzeile)
```

- **SMTP**

```
HELO cassini
```

```
MAIL FROM: <example@example.com>
```

```
RCPT TO: <beispiel@example.de>
```

```
(E-Mail-Header – Teil der Nutzdaten)
```

```
From: Eddie Example <example@example.com>
```

```
To: Bert Beispiel <beispiel@example.de>
```

```
Subject: Hello, world!
```

```
(Leerzeile)
```

```
Hi, there!
```

```
.
```

## 3.4 TCP-Protokolle

- **HTTP**

```
GET / HTTP/1.1  
Host: www.hs-bochum.de  
(Leerzeile)
```

- **SMTP**

```
HELO cassini  
MAIL FROM: <example@example.com>  
RCPT TO: <beispiel@example.de>  
(E-Mail-Header – Teil der Nutzdaten)  
From: Eddie Example <example@example.com>  
To: Bert Beispiel <beispiel@example.de>  
Subject: Hello, world!  
(Leerzeile)  
Hi, there!  
.
```

- Protokolle „mal eben“ selbst schreiben: `nc -c` oder `inetd`

## 3.5 Routing

- `ip route` (Linux)  
`route` (MS-Windows, Unix)  
`netstat -nr` (MacOS)

```
# route -n
```

```
Kernel-IP-Routentabelle
```

Ziel	Router	Genmask	[...]	Iface
0.0.0.0	192.168.42.1	0.0.0.0	[...]	wlan0
169.254.0.0	0.0.0.0	255.255.0.0	[...]	wlan0
192.168.42.0	0.0.0.0	255.255.255.0	[...]	wlan0

Netzmaske:

Wenn nach Und-Verknüpfung mit IP-Adresse gleich, → im gleichen Netz

255.255.240.0 ist dasselbe wie /20

(20 Bit sind 1; die restlichen 12 Bit sind 0)

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

13. November 2024



# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

## 2 Kurzeinführung Unix

## 3 Kurzeinführung TCP/IP

### 3.1 MAC-Adressen

### 3.2 IP-Adressen

### 3.3 TCP- und UDP-Ports

### 3.4 Protokolle

### 3.5 Routing

### 3.6 Netzwerkanalyse

### 3.7 SSH

### 3.8 X11

## 4 Relationale Datenbanken

...



Änderungen  
vorbehalten

## 3 Kurzeinführung TCP/IP

### 3.1 MAC-Adressen

MAC = Media Access Control

MAC-Adresse = Hardware-Adresse = Ethernet-Adresse

- `ip neigh`  
`arp`

## 3.2 IP-Adressen

- `ip addr` (Linux)  
  `ifconfig` (Unix allgemein)  
  `ipconfig` (MS Windows)
- `ip addr add <Netz>`
- `ip link`
- `ping <IP-Adresse>`

```
# ifconfig
```

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    [...]
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.42.101  netmask 255.255.255.0
                                broadcast 192.168.42.255
    ether be:3f:ca:aa:7e:51 txqueuelen 1000  (Ethernet)
    [...]
```

## 3.2 IP-Adressen

IPv4-Adressen:

- 32 Bit
- dezimal, 4 Gruppen zu je 8 Bit (0–255), durch Punkte getrennt

IPv6-Adressen:

- 128 Bit
- hexadezimal, 8 Gruppen zu je 4 Hex-Ziffern, durch Doppelpunkte getrennt
- Führende Nullen dürfen weggelassen werden.
- Zwei Doppelpunkte bedeuten: Mit Nullen auffüllen.
- Literatur und Beispiel: <https://de.wikipedia.org/wiki/IPv6>

## 3.3 TCP- und UDP-Ports

- `nc <IP> <Port>`  
Verbindung zu Programm <Port> auf Rechner <IP> aufnehmen
- `nc -l <Port>` oder `nc -p <Port> -l`  
auf eingehende Verbindungen warten („lauschen“)
- `nc -c <Programm>`  
Programm statt Mensch kommunizieren lassen
- TCP-Ports: Verbindungskonzept, Netzwerk prüft
- UDP-Ports: einzelne Pakete, Anwendung muß selbst prüfen
- ICMP: keine Ports, nur Rechner:  
Erreichbarkeit, Eigenschaften der Übertragung

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse

## 3.4 Protokolle

### TCP-Protokolle

- **HTTP**

```
GET / HTTP/1.1
```

```
Host: www.hs-bochum.de
```

```
(Leerzeile)
```

- **SMTP**

```
HELO cassini
```

```
MAIL FROM: <example@example.com>
```

```
RCPT TO: <beispiel@example.de>
```

```
(E-Mail-Header – Teil der Nutzdaten)
```

```
From: Eddie Example <example@example.com>
```

```
To: Bert Beispiel <beispiel@example.de>
```

```
Subject: Hello, world!
```

```
(Leerzeile)
```

```
Hi, there!
```

```
.
```

- Protokolle „mal eben“ selbst schreiben: `nc -c` oder `inetd`

## 3.4 Protokolle

### UDP-Protokolle

- **DNS**

Schnelle Abfrage von Informationen, insbesondere:  
Die zu einem Rechner-Namen gehörige IP-Adresse herausfinden

- **DHCP**

benutzbare IP-Adresse (und weitere Informationen) bei einem Server erfragen

### ICMP-Protokolle

- Informationsaustausch zwischen Rechnern zur Verwaltung des Netzwerks selbst
- Beispiel: `ping`

## 3.5 Routing

- `ip route` (Linux)  
`route` (MS-Windows, Unix)  
`netstat -nr` (MacOS)

```
# route -n
```

```
Kernel-IP-Routentabelle
```

Ziel	Router	Genmask	[...]	Iface
0.0.0.0	192.168.42.1	0.0.0.0	[...]	wlan0
169.254.0.0	0.0.0.0	255.255.0.0	[...]	wlan0
192.168.42.0	0.0.0.0	255.255.255.0	[...]	wlan0

Netzmaske:

Wenn nach Und-Verknüpfung mit IP-Adresse gleich, → im gleichen Netz

255.255.240.0 ist dasselbe wie /20

(20 Bit sind 1; die restlichen 12 Bit sind 0)



## 3.6 Netzwerkanalyse

- `tcpdump`
- `wireshark`
- `ettercap`

## 3.7 SSH

- SSH <Rechner>
- -C: Komprimierung
- -L: lokalen Port auf Remote-Port umleiten
- -R: Remote-Port auf lokalen Port umleiten

## 3.8 X11

- Grafik-Bildschirm und Eingabegeräte über's Netz
- `DISPLAY`-Variable: X-Server: Rechner und Bildschirm
- `ssh -X`: X11-Forwarding

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

20. November 2024

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

## 2 Kurzeinführung Unix

## 3 Kurzeinführung TCP/IP

### 3.1 MAC-Adressen

### 3.2 IP-Adressen

### 3.3 TCP- und UDP-Ports

### 3.4 Protokolle

### 3.5 Routing

### 3.6 Netzwerkanalyse

### 3.7 SSH

### 3.8 X11 und VNC

### 3.9 GNU screen

### 3.10 Firewall

## 4 Relationale Datenbanken

...



Änderungen  
vorbehalten

## 3.2 IP-Adressen

IPv4-Adressen:

- 32 Bit
- dezimal, 4 Gruppen zu je 8 Bit (0–255), durch Punkte getrennt

IPv6-Adressen:

- 128 Bit
- hexadezimal, 8 Gruppen zu je 4 Hex-Ziffern, durch Doppelpunkte getrennt
- Führende Nullen dürfen weggelassen werden.
- Zwei Doppelpunkte bedeuten: Mit Nullen auffüllen.
- Literatur und Beispiel: <https://de.wikipedia.org/wiki/IPv6>

## 3.3 TCP- und UDP-Ports

- `nc <IP> <Port>`  
Verbindung zu Programm <Port> auf Rechner <IP> aufnehmen
- `nc -l <Port>` oder `nc -p <Port> -l`  
auf eingehende Verbindungen warten („lauschen“)
- `nc -c <Programm>`  
Programm statt Mensch kommunizieren lassen
- TCP-Ports: Verbindungskonzept, Netzwerk prüft
- UDP-Ports: einzelne Pakete, Anwendung muß selbst prüfen
- ICMP: keine Ports, nur Rechner:  
Erreichbarkeit, Eigenschaften der Übertragung

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse

## 3.4 Protokolle

### TCP-Protokolle

- **HTTP**

GET / HTTP/1.1

Host: www.hs-bochum.de

(Leerzeile)

- **SMTP**

HELO cassini

MAIL FROM: <example@example.com>

RCPT TO: <beispiel@example.de>

(E-Mail-Header – Teil der Nutzdaten)

From: Eddie Example <example@example.com>

To: Bert Beispiel <beispiel@example.de>

Subject: Hello, world!

(Leerzeile)

Hi, there!

.

- Protokolle „mal eben“ selbst schreiben: `nc -c`  
oder `man systemd.socket` oder (veraltet) `inetd`



## 3.4 Protokolle

### UDP-Protokolle

- Video- und Audio-Übertragung (z. B. `mumble`)
- **DNS** (auch TCP)  
Schnelle Abfrage von Informationen, insbesondere:  
Die zu einem Rechner-Namen gehörige IP-Adresse herausfinden
- **DHCP**  
benutzbare IP-Adresse (und weitere Informationen) bei einem Server erfragen

### ICMP-Protokolle

- Informationsaustausch zwischen Rechnern zur Verwaltung des Netzwerks selbst
- Beispiel: `ping`

## 3.5 Routing

- `ip route` (Linux)  
`route` (Unix)  
`route print` (MS-Windows)  
`netstat -nr` (Linux, MS-Windows, MacOS)

```
# route -n
```

```
Kernel-IP-Routentabelle
```

Ziel	Router	Genmask	[...]	Iface
0.0.0.0	192.168.42.1	0.0.0.0	[...]	wlan0
169.254.0.0	0.0.0.0	255.255.0.0	[...]	wlan0
192.168.42.0	0.0.0.0	255.255.255.0	[...]	wlan0

Netzmaske:

Wenn nach Und-Verknüpfung mit IP-Adresse gleich, → im gleichen Netz

255.255.240.0 ist dasselbe wie /20

(20 Bit sind 1; die restlichen 12 Bit sind 0)

## 3.6 Netzwerkanalyse

- `tcpdump`
- `wireshark` (GUI), `tshark` (CLI)
- `ettercap`

## 3.7 SSH

- SSH <Rechner>
- -C: Komprimierung
- -L: lokalen Port auf Remote-Port umleiten
- -R: Remote-Port auf lokalen Port umleiten

## 3.8 X11 und VNC

- Grafik-Bildschirm und Eingabegeräte über's Netz
- `DISPLAY`-Variable: X-Server: Rechner und Bildschirm
- `ssh -X`, `ssh -Y`: X11-Forwarding

Warnung: Der entfernte Rechner bekommt Zugriff auf den X-Server und kann insbesondere Tastatureingaben mitlesen!

Für Details siehe: X11-SECURITY-Erweiterungen:

<https://www.x.org/wiki/Development/Documentation/Security/>

## 3.8 X11 und VNC

- Grafik-Bildschirm und Eingabegeräte über's Netz
- `DISPLAY`-Variable: X-Server: Rechner und Bildschirm
- `ssh -X`, `ssh -Y`: X11-Forwarding

Warnung: Der entfernte Rechner bekommt Zugriff auf den X-Server und kann insbesondere Tastatureingaben mitlesen!

Für Details siehe: X11-SECURITY-Erweiterungen:

<https://www.x.org/wiki/Development/Documentation/Security/>

- VNC = Virtual Network Computing
- VNC-Server stellt Bildschirminhalt zur Verfügung
  - entweder: eigener, virtueller X11-Server
  - oder: ruft Inhalt von anderem (X11-) Bildschirm ab
- VNC-Client ruft Bildschirminhalt ab und stellt ihn dar
  - z. B. per X11
  - z. B. per Web-Interface: noVNC

## 3.9 GNU screen

- Text-Bildschirm und Eingabegeräte über's Netz
- `Ctrl+A c`: neues Fenster (create)
- `Ctrl+A Leertaste`: nächstes Fenster
- `Ctrl+A 3`: Fenster Nr. 3
- `Ctrl+A ESC`: hochscrollen, suchen, markieren (copy)
- `Ctrl+A ]`: einfügen (paste)
- `Ctrl+A d`: von `screen` ablösen (detach)
- `screen -x`: an laufenden `screen` andocken, auch geteilt
- Ablösen/Andocken, auch geteilt, für Grafik: `VNC`, `x2go`

Alternative: `tmux`

## 3.10 Firewall

- ~~magische Hardware und/oder Software,  
die uns vor allen Angriffen schützt~~
- Paketfilter (Internet- und Transportschicht: IP-Adressen, Port-Nummern)
- ...

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse



# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

27. November 2024

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

## 2 Kurzeinführung Unix

## 3 Kurzeinführung TCP/IP

### 3.1 MAC-Adressen

### 3.2 IP-Adressen

### 3.3 TCP- und UDP-Ports

### 3.4 Protokolle

### 3.5 Routing

### 3.6 Netzwerkanalyse

### 3.7 SSH

### 3.8 X11 und VNC

### 3.9 GNU screen

### 3.10 Firewall

## 4 Relationale Datenbanken

...



Änderungen  
vorbehalten

## 3.7 SSH

- Sicheres<sup>1</sup> Arbeiten auf einem entfernten Rechner
- `ssh <Rechner>`
- `-C`: Komprimierung
- `-L`: lokalen Port auf Remote-Port umleiten
- `-R`: Remote-Port auf lokalen Port umleiten
  
- Beim ersten Verbinden:  
öffentlichem Schlüssel des entfernten Rechners prüfen  
(mit `ssh-keygen` abrufen, wird in `.ssh/known_hosts` hinterlegt)
- Anstelle eines Passwortes: eigenes Schlüsselpaar erzeugen,  
öffentlichem Schlüssel in `.ssh/authorized_keys` hinterlegen
- Zusätzliche Sicherheit:  
eigenen geheimen Schlüssel durch Passwort schützen
- Passwort nicht jedes Mal neu eingeben:  
SSH-Agent (`ssh-add`)

---

<sup>1</sup>gemäß Stand der Technik, so sicher wie z. Zt. überhaupt möglich

## 3.8 X11 und VNC

- Grafik-Bildschirm und Eingabegeräte über's Netz
- `DISPLAY`-Variable: X-Server: Rechner und Bildschirm
- `ssh -X`, `ssh -Y`: X11-Forwarding

Warnung: Der entfernte Rechner bekommt Zugriff auf den X-Server und kann insbesondere Tastatureingaben mitlesen!

Für Details siehe: X11-SECURITY-Erweiterungen:

<https://www.x.org/wiki/Development/Documentation/Security/>

- VNC = Virtual Network Computing
- VNC-Server stellt Bildschirminhalt zur Verfügung
  - entweder: eigener, virtueller X11-Server
  - oder: ruft Inhalt von anderem (X11-) Bildschirm ab
- VNC-Client ruft Bildschirminhalt ab und stellt ihn dar
  - z. B. per X11
  - z. B. per Web-Interface: noVNC

## 3.9 GNU screen

- Text-Bildschirm und Eingabegeräte über's Netz
- `Ctrl+A c`: neues Fenster (create)
- `Ctrl+A Leertaste`: nächstes Fenster
- `Ctrl+A 3`: Fenster Nr. 3
- `Ctrl+A ESC`: hochscrollen, suchen, markieren (copy)
- `Ctrl+A ]`: einfügen (paste)
- `Ctrl+A d`: von `screen` ablösen (detach)
- `screen -x`: an laufenden `screen` andocken, auch geteilt
- Ablösen/Andocken, auch geteilt, für Grafik: `VNC`, `x2go`
- Alternative: `tmux`

## 3.10 Firewall

- ~~magische Hardware und/oder Software,  
die uns vor allen Angriffen schützt~~
- Paketfilter (Internet- und Transportschicht: IP-Adressen, Port-Nummern)

## 3.10 Firewall

- ~~magische Hardware und/oder Software,  
die uns vor allen Angriffen schützt~~
  - Paketfilter (Internet- und Transportschicht: IP-Adressen, Port-Nummern)
    - Stateful Packet Inspection (Transportschicht)
    - Deep Packet Inspection (Anwendungsschicht)
  - Proxy (Anwendungsschicht)
    - Anfragen verstehen,  
stellvertretend für die Anwendung beim Server anfragen
- > mehrere Anfragen bündeln: weniger Netzlast
- > Inhalte filtern: Malware erkennen,  
aber auch „politisch unerwünschte“ Inhalte

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse

# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
root@cassini:~# su - postgres
postgres@cassini:~$ psql
psql (15.3 (Debian 15.3-0+deb12u1))
Geben Sie »help« für Hilfe ein.
```

```
postgres=# help
Dies ist psql, die Kommandozeilenschnittstelle für PostgreSQL.
Geben Sie ein:  \copyright für Urheberrechtsinformationen
                \h für Hilfe über SQL-Anweisungen
                \? für Hilfe über interne Anweisungen
                \g oder Semikolon, um eine Anfrage auszuführen
                \q um zu beenden

postgres=#
```



# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
postgres=# \?  
Allgemein  
  \copyright           PostgreSQL-Urheberrechtsinformationen zeigen  
  \crosstabview [SPALTEN] Anfrage ausführen und Ergebnis als Kreuztabelle  
                        anzeigen  
  \errverbose          letzte Fehlermeldung mit vollen Details anzeigen  
  \g [(OPT)] [DATEI]  SQL-Anweisung ausführen (und Ergebnis in Datei  
                        oder |Pipe schreiben); ...  
  ...  
  \q                  psql beenden  
  ...
```

```
postgres=#
```

# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
postgres=# \l
```

Liste der Datenbanken					
Name	Eigentümer	Kodierung	Sortierfolge	Zeichentyp	
postgres	postgres	UTF8	de_DE.UTF-8	de_DE.UTF-8	
template0	postgres	UTF8	de_DE.UTF-8	de_DE.UTF-8	
template1	postgres	UTF8	de_DE.UTF-8	de_DE.UTF-8	
(3 Zeilen)					

```
postgres=#
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
postgres=# create database testdb;  
CREATE DATABASE  
postgres=# create user dbsadmin with password '####';  
CREATE ROLE  
postgres=# create user dbs with password '####';  
CREATE ROLE  
postgres=# ALTER DATABASE testdb OWNER TO dbsadmin;  
ALTER DATABASE  
postgres=# \q  
postgres@cassini:~$
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
$ psql -h localhost -U dbadmin -W testdb
```

```
Password: ####
```

```
psql (15.5 (Debian 15.5-0+deb12u1))
```

```
SSL-Verbindung (Protokoll: TLSv1.3, Verschlüsselungsmethode: TLS_AES_256_G
```

```
Geben Sie »help« für Hilfe ein.
```

```
testdb=> grant select, insert, update on tier to dbs;
```

```
GRANT
```

```
testdb=> \q
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
$ psql -h localhost -U dbs -W testdb
Passwort: ####
psql (15.5 (Debian 15.5-0+deb12u1))
SSL-Verbindung (Protokoll: TLSv1.3, Verschlüsselungsmethode: TLS_AES_256_G
Geben Sie »help« für Hilfe ein.
```

```
testdb=> drop table tier;
FEHLER:  Berechtigung nur für Eigentümer der Tabelle tier
testdb=>
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

Literatur: z. B. [https://de.wikibooks.org/wiki/Einführung\\_in\\_SQL](https://de.wikibooks.org/wiki/Einführung_in_SQL)

Wichtige SQL-Befehle:

- **CREATE** – Datenbanken, Tabellen usw. anlegen
- **DROP** – Datenbanken, Tabellen usw. löschen
- **SELECT** – Daten abfragen
- **INSERT INTO ... VALUES** – Daten eingeben
- **UPDATE** – Daten ändern
- **DELETE FROM** – Daten löschen

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

4. Dezember 2024

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

## 2 Kurzeinführung Unix

## 3 Kurzeinführung TCP/IP

...

### 3.10 Firewall

## 4 Relationale Datenbanken

### 4.1 Einführung in DBMS

### 4.2 Einführung in SQL

### 4.3 Normalformen

### 4.4 Verknüpfungen von Tabellen

### 4.5 Sichten

### 4.6 Schlüsselfelder

### 4.7 Datensicherung

...



Änderungen  
vorbehalten



## 3.10 Firewall

- ~~magische Hardware und/oder Software,  
die uns vor allen Angriffen schützt~~
  - Paketfilter (Internet- und Transportschicht: IP-Adressen, Port-Nummern)
    - Stateful Packet Inspection (Transportschicht)
    - Deep Packet Inspection (Anwendungsschicht)
  - Proxy (Anwendungsschicht)
    - Anfragen verstehen,  
stellvertretend für die Anwendung beim Server anfragen
- > mehrere Anfragen bündeln: weniger Netzlast
- > Inhalte filtern: Malware erkennen,  
aber auch „politisch unerwünschte“ Inhalte

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse

# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
root@cassini:~# su - postgres
postgres@cassini:~$ psql
psql (15.3 (Debian 15.3-0+deb12u1))
Geben Sie »help« für Hilfe ein.
```

```
postgres=# help
Dies ist psql, die Kommandozeilenschnittstelle für PostgreSQL.
Geben Sie ein:  \copyright für Urheberrechtsinformationen
                \h für Hilfe über SQL-Anweisungen
                \? für Hilfe über interne Anweisungen
                \g oder Semikolon, um eine Anfrage auszuführen
                \q um zu beenden

postgres=#
```

# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
postgres=# \?  
Allgemein  
  \copyright           PostgreSQL-Urheberrechtsinformationen zeigen  
  \crosstabview [SPALTEN] Anfrage ausführen und Ergebnis als Kreuztabelle  
                        anzeigen  
  \errverbose          letzte Fehlermeldung mit vollen Details anzeigen  
  \g [(OPT)] [DATEI]   SQL-Anweisung ausführen (und Ergebnis in Datei  
                        oder |Pipe schreiben); ...  
  ...                  ...  
  \q                   psql beenden  
  ...                  ...
```

```
postgres=#
```

# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
postgres=# \l
```

```

                                Liste der Datenbanken
  Name      | Eigentümer | Kodierung | Sortierfolge | Zeichentyp | ...
-----+-----+-----+-----+-----+-----
 postgres   | postgres   | UTF8      | de_DE.UTF-8  | de_DE.UTF-8 |
 template0  | postgres   | UTF8      | de_DE.UTF-8  | de_DE.UTF-8 |
            |            |           |              |              |
 template1  | postgres   | UTF8      | de_DE.UTF-8  | de_DE.UTF-8 |
            |            |           |              |              |
(3 Zeilen)
```

```
postgres=#
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
postgres=# create database testdb;  
CREATE DATABASE  
postgres=# create user dbsadmin with password '####';  
CREATE ROLE  
postgres=# create user dbs with password '####';  
CREATE ROLE  
postgres=# ALTER DATABASE testdb OWNER TO dbsadmin;  
ALTER DATABASE  
postgres=# \q  
postgres@cassini:~$
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
$ psql -h localhost -U dbadmin -W testdb
```

```
Password: ####
```

```
psql (15.5 (Debian 15.5-0+deb12u1))
```

```
SSL-Verbindung (Protokoll: TLSv1.3, Verschlüsselungsmethode: TLS_AES_256_G
```

```
Geben Sie »help« für Hilfe ein.
```

```
testdb=> grant select, insert, update on tier to dbs;
```

```
GRANT
```

```
testdb=> \q
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
$ psql -h localhost -U dbs -W testdb
Passwort: ####
psql (15.5 (Debian 15.5-0+deb12u1))
SSL-Verbindung (Protokoll: TLSv1.3, Verschlüsselungsmethode: TLS_AES_256_G
Geben Sie »help« für Hilfe ein.
```

```
testdb=> drop table tier;
FEHLER:  Berechtigung nur für Eigentümer der Tabelle tier
testdb=>
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

Literatur: z. B. [https://de.wikibooks.org/wiki/Einführung\\_in\\_SQL](https://de.wikibooks.org/wiki/Einführung_in_SQL)

Wichtige SQL-Befehle:

- **CREATE** – Datenbanken, Tabellen usw. anlegen
- **DROP** – Datenbanken, Tabellen usw. löschen
- **SELECT** – Daten abfragen
- **INSERT INTO ... VALUES** – Daten eingeben
- **UPDATE** – Daten ändern
- **DELETE FROM** – Daten löschen



# 4 Relationale Datenbanken

## 4.3 Normalformen

Problem: Schlecht angelegte Datenbanken werden schnell inkonsistent.  
Beliebte Fehler:

- Speichern von mehreren Daten in demselben Tabelleneintrag
- Speichern von denselben Daten in verschiedenen Tabelleneinträgen
- implizite Zusammenhänge
- voneinander unabhängige Zusammenhänge in derselben Tabelle

Literatur: z. B. [https://de.wikipedia.org/wiki/Normalisierung\\_\(Datenbank\)](https://de.wikipedia.org/wiki/Normalisierung_(Datenbank))

# 4 Relationale Datenbanken

## 4.3 Normalformen

Problem: Schlecht angelegte Datenbanken werden schnell inkonsistent.  
Beliebte Fehler:

- Speichern von mehreren Daten in demselben Tabelleneintrag  
→ 1. Normalform
- Speichern von denselben Daten in verschiedenen Tabelleneinträgen  
→ 2. Normalform
- implizite Zusammenhänge  
→ 3. Normalform und Boyce-Codd-Normalform
- voneinander unabhängige Zusammenhänge in derselben Tabelle  
→ 4. und 5. Normalform

Lösung: Normalformen

Literatur: z. B. [https://de.wikipedia.org/wiki/Normalisierung\\_\(Datenbank\)](https://de.wikipedia.org/wiki/Normalisierung_(Datenbank))

## 4 Relationale Datenbanken

### 4.4 Verknüpfungen von Tabellen

Problem: Gut angelegte Datenbanken (→ Normalformen) sind stark aufgesplittet.

Wie kann man sie weiterhin effizient benutzen?

Lösung: Verknüpfungen von Tabellen

SQL-Befehl: JOIN

Literatur: z. B. <https://de.wikipedia.org/wiki/SQL>

## 4 Relationale Datenbanken

### 4.4 Verknüpfungen von Tabellen

Was machen wir mit Tabelleneinträgen,  
bei denen die **ON**-Bedingung nicht erfüllt ist?

```
SELECT <Feld[er]> FROM <Tabelle1> [INNER] JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ weglassen: **[INNER] JOIN**

```
SELECT <Feld[er]> FROM <Tabelle1> LEFT JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ linke Tabelle trotzdem anzeigen (mit **NULL**-Einträgen): **LEFT JOIN**  
(analog: **RIGHT JOIN** für rechte Tabelle)

```
SELECT <Feld[er]> FROM <Tabelle1> FULL JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ beide Tabellen trotzdem anzeigen (mit **NULL**-Einträgen): **FULL JOIN**

## 4 Relationale Datenbanken

### 4.5 Sichten

```
SELECT <Feld[er]> FROM <Tabelle1> [INNER] JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ Wir betrachten beide Tabellen zusammen als eine große Tabelle.

```
CREATE VIEW <Sicht> AS  
    SELECT <Feld[er]> FROM <Tabelle1> JOIN <Tabelle2>  
        ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;  
SELECT <Feld[er]> FROM <Sicht> [WHERE ...];
```

→ Wir sprechen das Ergebnis genau wie eine Tabelle an.

→ Es ist möglich, ohne Verlust an Komfort alle Daten in Normalform zu halten.

# 4 Relationale Datenbanken

## 4.6 Schlüsselfelder

```
CREATE TABLE tabelle1 (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    ...  
);  
CREATE TABLE tabelle2 (  
    ...  
    tabelle1_id INT,  
    ...  
    FOREIGN KEY(tabelle1_id) REFERENCES tabelle1(id)  
);
```

- Dem DBMS mitteilen, welche Felder für **JOIN** vorgesehen sind.
- Das DBMS kann mit auf Konsistenz achten.

## 4 Relationale Datenbanken

### 4.7 Datensicherung

```
$ pg_dump --clean -h <Rechner> -U <User> -W <Datenbank>
```

- Ausgabe des gesamten Datenbankinhalts als SQL-Quelltext zur Standardausgabe

—> keine Probleme mit sich evtl. ändernden Binärformaten

- Es ist möglich, den Inhalt direkt in einer Pipe weiterzuverarbeiten (z. B. zu komprimieren).
- Zurückspielen: mit `psql`

```
$ psql -h <Rechner> -U <User> -W <Datenbank> \  
    < <Ausgabe von pg_dump>
```

- analog für **MariaDB**: `mariadb-dump`

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

11. Dezember 2024



# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

- 1 Einführung
- 2 Kurzeinführung Unix
- 3 Kurzeinführung TCP/IP
- 4 Relationale Datenbanken
  - 4.1 Einführung in DBMS
  - 4.2 Einführung in SQL
  - 4.3 Normalformen
  - 4.4 Verknüpfungen von Tabellen
  - 4.5 Sichten
  - 4.6 Schlüsselfelder
  - 4.7 Datensicherung
  - 4.8 Aggregate
  - 4.9 Transaktionen
  - 4.10 Indizierung
  - 4.11 Funktionen und Trigger

...



Änderungen  
vorbehalten

# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
root@cassini:~# su - postgres
postgres@cassini:~$ psql
psql (15.3 (Debian 15.3-0+deb12u1))
Geben Sie »help« für Hilfe ein.
```

```
postgres=# help
Dies ist psql, die Kommandozeilenschnittstelle für PostgreSQL.
Geben Sie ein:  \copyright für Urheberrechtsinformationen
                \h für Hilfe über SQL-Anweisungen
                \? für Hilfe über interne Anweisungen
                \g oder Semikolon, um eine Anfrage auszuführen
                \q um zu beenden

postgres=#
```

# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
postgres=# \?  
Allgemein  
  \copyright           PostgreSQL-Urheberrechtsinformationen zeigen  
  \crosstabview [SPALTEN] Anfrage ausführen und Ergebnis als Kreuztabelle  
                        anzeigen  
  \errverbose          letzte Fehlermeldung mit vollen Details anzeigen  
  \g [(OPT)] [DATEI]   SQL-Anweisung ausführen (und Ergebnis in Datei  
                        oder |Pipe schreiben); ...  
  ...                  ...  
  \q                   psql beenden  
  ...                  ...
```

```
postgres=#
```

# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
postgres=# \l
```

Liste der Datenbanken					
Name	Eigentümer	Kodierung	Sortierfolge	Zeichentyp	...
postgres	postgres	UTF8	de_DE.UTF-8	de_DE.UTF-8	
template0	postgres	UTF8	de_DE.UTF-8	de_DE.UTF-8	
template1	postgres	UTF8	de_DE.UTF-8	de_DE.UTF-8	
(3 Zeilen)					...

```
postgres=#
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
postgres=# create database testdb;  
CREATE DATABASE  
postgres=# create user dbsadmin with password '####';  
CREATE ROLE  
postgres=# create user dbs with password '####';  
CREATE ROLE  
postgres=# ALTER DATABASE testdb OWNER TO dbsadmin;  
ALTER DATABASE  
postgres=# \q  
postgres@cassini:~$
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
$ psql -h localhost -U dbadmin -W testdb
```

```
Password: ####
```

```
psql (15.5 (Debian 15.5-0+deb12u1))
```

```
SSL-Verbindung (Protokoll: TLSv1.3, Verschlüsselungsmethode: TLS_AES_256_G
```

```
Geben Sie »help« für Hilfe ein.
```

```
testdb=> grant select, insert, update on tier to dbs;
```

```
GRANT
```

```
testdb=> \q
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
$ psql -h localhost -U dbs -W testdb
Passwort: ####
psql (15.5 (Debian 15.5-0+deb12u1))
SSL-Verbindung (Protokoll: TLSv1.3, Verschlüsselungsmethode: TLS_AES_256_G
Geben Sie »help« für Hilfe ein.
```

```
testdb=> drop table tier;
FEHLER: Berechtigung nur für Eigentümer der Tabelle tier
testdb=>
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

Literatur: z. B. [https://de.wikibooks.org/wiki/Einführung\\_in\\_SQL](https://de.wikibooks.org/wiki/Einführung_in_SQL)

Wichtige SQL-Befehle:

- **CREATE** – Datenbanken, Tabellen usw. anlegen
- **DROP** – Datenbanken, Tabellen usw. löschen
- **SELECT** – Daten abfragen  
auswählen mit **WHERE**, sortieren mit **ORDER BY**
- **INSERT INTO ... VALUES** – Daten eingeben
- **UPDATE** – Daten ändern
- **DELETE FROM** – Daten löschen

(Manche DBMS akzeptieren nur Großbuchstaben.)



# 4 Relationale Datenbanken

## 4.3 Normalformen

Problem: Schlecht angelegte Datenbanken werden schnell inkonsistent.  
Beliebte Fehler:

- Speichern von mehreren Daten in demselben Tabelleneintrag
- Speichern von denselben Daten in verschiedenen Tabelleneinträgen
- implizite Zusammenhänge
- voneinander unabhängige Zusammenhänge in derselben Tabelle

Literatur: z. B. [https://de.wikipedia.org/wiki/Normalisierung\\_\(Datenbank\)](https://de.wikipedia.org/wiki/Normalisierung_(Datenbank))

# 4 Relationale Datenbanken

## 4.3 Normalformen

Problem: Schlecht angelegte Datenbanken werden schnell inkonsistent.  
Beliebte Fehler:

- Speichern von mehreren Daten in demselben Tabelleneintrag  
→ 1. Normalform
- Speichern von denselben Daten in verschiedenen Tabelleneinträgen  
→ 2. Normalform
- implizite Zusammenhänge  
→ 3. Normalform und Boyce-Codd-Normalform
- voneinander unabhängige Zusammenhänge in derselben Tabelle  
→ 4. und 5. Normalform

Lösung: Normalformen

Literatur: z. B. [https://de.wikipedia.org/wiki/Normalisierung\\_\(Datenbank\)](https://de.wikipedia.org/wiki/Normalisierung_(Datenbank))

## 4 Relationale Datenbanken

### 4.4 Verknüpfungen von Tabellen

Problem: Gut angelegte Datenbanken (→ Normalformen) sind stark aufgesplittet.

Wie kann man sie weiterhin effizient benutzen?

Lösung: Verknüpfungen von Tabellen

SQL-Befehl: JOIN

Literatur: z. B. <https://de.wikipedia.org/wiki/SQL>

## 4 Relationale Datenbanken

### 4.4 Verknüpfungen von Tabellen

Was machen wir mit Tabelleneinträgen,  
bei denen die **ON**-Bedingung nicht erfüllt ist?

```
SELECT <Feld[er]> FROM <Tabelle1> [INNER] JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ weglassen: **[INNER] JOIN**

```
SELECT <Feld[er]> FROM <Tabelle1> LEFT JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ linke Tabelle trotzdem anzeigen (mit **NULL**-Einträgen): **LEFT JOIN**  
(analog: **RIGHT JOIN** für rechte Tabelle)

```
SELECT <Feld[er]> FROM <Tabelle1> FULL JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ beide Tabellen trotzdem anzeigen (mit **NULL**-Einträgen): **FULL JOIN**

## 4 Relationale Datenbanken

### 4.5 Sichten

```
SELECT <Feld[er]> FROM <Tabelle1> [INNER] JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ Wir betrachten beide Tabellen zusammen als eine große Tabelle.

```
CREATE VIEW <Sicht> AS  
    SELECT <Feld[er]> FROM <Tabelle1> JOIN <Tabelle2>  
        ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;  
SELECT <Feld[er]> FROM <Sicht> [WHERE ...];
```

→ Wir sprechen das Ergebnis genau wie eine Tabelle an.

→ Es ist möglich, ohne Verlust an Komfort alle Daten in Normalform zu halten.

# 4 Relationale Datenbanken

## 4.6 Schlüsselfelder

```
CREATE TABLE tabelle1 (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    ...  
);  
CREATE TABLE tabelle2 (  
    ...  
    tabelle1_id INT,  
    ...  
    FOREIGN KEY(tabelle1_id) REFERENCES tabelle1(id)  
);
```

→ Dem DBMS mitteilen, welche Felder für **JOIN** vorgesehen sind.

→ Das DBMS kann mit auf Konsistenz achten.

## 4 Relationale Datenbanken

### 4.7 Datensicherung

```
$ pg_dump --clean -h <Rechner> -U <User> -W <Datenbank>
```

- Ausgabe des gesamten Datenbankinhalts als SQL-Quelltext zur Standardausgabe

—> keine Probleme mit sich evtl. ändernden Binärformaten

- Es ist möglich, den Inhalt direkt in einer Pipe weiterzuverarbeiten (z. B. zu komprimieren).
- Zurückspielen: mit `psql`

```
$ psql -h <Rechner> -U <User> -W <Datenbank> \  
    < <Ausgabe von pg_dump>
```

- analog für **MariaDB**: `mariadb-dump`

# 4 Relationale Datenbanken

## 4.8 Aggregate

Rechenoperationen auf der ganzen Spalte, z. B. Summenbildung:

```
testdb=> SELECT SUM (id) FROM tier;
```

```
sum
```

```
-----
```

```
4143
```

Rechenoperationen: SUM, AVG, MIN, MAX, COUNT

Gruppierung:

```
testdb=> SELECT tierart, MIN (id) AS "Minimum" FROM tier
        GROUP BY tierart;
```

tierart	Minimum
Spinne	94
Hund	7
Kaninchen	42

Auswahl mit HAVING (entsprechend WHERE)



# 4 Relationale Datenbanken

## 4.9 Transaktionen

- Ziel: Wahrung der Konsistenz
- Methode: zusammengehörige Aktionen zu einer *Transaktion* zusammenfassen
- Beispiel: Überweisung  
Betrag von einem Konto subtrahieren  
und „gleichzeitig“ zu einem anderen Konto addieren
- Realisierung in PostgreSQL: `BEGIN; ... COMMIT;`
- Realisierung in MariaDB: `START TRANSACTION; ... COMMIT;`
- Abbruch einer Transaktion: `ROLLBACK;` statt `COMMIT;`

# 4 Relationale Datenbanken

## 4.10 Indizierung

- Ziel: Performanzgewinn
- Methode: Spalten, in denen häufig gesucht wird, sortieren
- Beispiel: Datenbank mit Kontaktdaten  
Suche nach Name, Adresse, Telefonnummer
- Realisierung in PostgreSQL und MariaDB:

```
CREATE INDEX <Indexname> ON <Tabellenname>
(
    <Spaltenname>,
    ...
);
```

- Vorteil: schnellere Suche (**SELECT**) in indizierten Spalten

# 4 Relationale Datenbanken

## 4.10 Indizierung

- Ziel: Performanzgewinn
- Methode: Spalten, in denen häufig gesucht wird, sortieren
- Beispiel: Datenbank mit Kontaktdaten  
Suche nach Name, Adresse, Telefonnummer
- Realisierung in PostgreSQL und MariaDB:

```
CREATE INDEX <Indexname> ON <Tabellenname>
(
    <Spaltenname>,
    ...
);
```

- Vorteil: schnellere Suche (**SELECT**) in indizierten Spalten
- Nachteil: langsames Einfügen/Ändern/Löschen

## 4 Relationale Datenbanken

### 4.11 Funktionen und Trigger

Funktionen:

- **PROCEDURE** entspricht einer **void**-Funktion in C.
- <https://www.postgresql.org/docs/15/sql-createprocedure.html>

Trigger:

- <https://www.sqltutorial.org/sql-triggers/>
- <https://www.postgresqltutorial.com/postgresql-triggers/creating-first-trigger-postgresql/>

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

18. Dezember 2024

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

- 1 Einführung
- 2 Kurzeinführung Unix
- 3 Kurzeinführung TCP/IP
- 4 Relationale Datenbanken
  - 4.1 Einführung in DBMS
  - 4.2 Einführung in SQL
  - 4.3 Normalformen
  - 4.4 Verknüpfungen von Tabellen
  - 4.5 Sichten
  - 4.6 Schlüsselfelder
  - 4.7 Datensicherung
  - 4.8 Aggregate
  - 4.9 Transaktionen
  - 4.10 Indizierung
  - 4.11 Funktionen und Trigger

...

...



Änderungen  
vorbehalten

# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
root@cassini:~# su - postgres
postgres@cassini:~$ psql
psql (15.3 (Debian 15.3-0+deb12u1))
Geben Sie »help« für Hilfe ein.
```

```
postgres=# help
Dies ist psql, die Kommandozeilenschnittstelle für PostgreSQL.
Geben Sie ein:  \copyright für Urheberrechtsinformationen
                 \h für Hilfe über SQL-Anweisungen
                 \? für Hilfe über interne Anweisungen
                 \g oder Semikolon, um eine Anfrage auszuführen
                 \q um zu beenden

postgres=#
```

# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
postgres=# \?  
Allgemein  
  \copyright           PostgreSQL-Urheberrechtsinformationen zeigen  
  \crosstabview [SPALTEN] Anfrage ausführen und Ergebnis als Kreuztabelle  
                        anzeigen  
  \errverbose          letzte Fehlermeldung mit vollen Details anzeigen  
  \g [(OPT)] [DATEI]   SQL-Anweisung ausführen (und Ergebnis in Datei  
                        oder |Pipe schreiben); ...  
  ...                  ...  
  \q                   psql beenden  
  ...                  ...  
  
postgres=#
```



# 4 Relationale Datenbanken

## 4.1 Einführung in DBMS

Datenbank-Management-System (DBMS): z. B. PostgreSQL, MariaDB

- Datenbanken „von außen“ verwalten (z. B. auflisten)
- Schnittstelle für Zugriff auf Datenbank
- **in hohem Maße herstellerspezifisch**

```
postgres=# \l
```

```

                        Liste der Datenbanken
  Name      | Eigentümer | Kodierung | Sortierfolge | Zeichentyp | ...
-----+-----+-----+-----+-----+
 postgres   | postgres   | UTF8      | de_DE.UTF-8  | de_DE.UTF-8 |
 template0  | postgres   | UTF8      | de_DE.UTF-8  | de_DE.UTF-8 |
            |            |           |              |              |
 template1  | postgres   | UTF8      | de_DE.UTF-8  | de_DE.UTF-8 |
            |            |           |              |              |
(3 Zeilen)
```

```
postgres=#
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
postgres=# CREATE DATABASE testdb;  
CREATE DATABASE  
postgres=# CREATE USER dbadmin WITH PASSWORD '####';  
CREATE ROLE  
postgres=# CREATE USER dbs WITH PASSWORD '####';  
CREATE ROLE  
postgres=# ALTER DATABASE testdb OWNER TO dbadmin;  
ALTER DATABASE  
postgres=# \q  
postgres@cassini:~$
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
$ psql -h localhost -U dbadmin -W testdb
```

```
Passwort: ####
```

```
psql (15.5 (Debian 15.5-0+deb12u1))
```

```
SSL-Verbindung (Protokoll: TLSv1.3, Verschlüsselungsmethode: TLS_AES_256_G
```

```
Geben Sie »help« für Hilfe ein.
```

```
testdb=> CREATE TABLE tier ( name TEXT, tierart TEXT, id INTEGER );
```

```
CREATE
```

```
testdb=> GRANT SELECT, INSERT, UPDATE, DELETE ON tier TO dbs;
```

```
GRANT
```

```
testdb=> \q
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

- Datenbanken „von innen“ verwalten, z. B. anlegen und wieder löschen
- Daten in der Datenbank bearbeiten
- **in hohem Maße standardisiert**

```
$ psql -h localhost -U dbs -W testdb
Passwort: ####
psql (15.5 (Debian 15.5-0+deb12u1))
SSL-Verbindung (Protokoll: TLSv1.3, Verschlüsselungsmethode: TLS_AES_256_G
Geben Sie »help« für Hilfe ein.
```

```
testdb=> DROP TABLE tier;
FEHLER:  Berechtigung nur für Eigentümer der Tabelle tier
testdb=>
```

# 4 Relationale Datenbanken

## 4.2 Einführung in SQL

Datenbank-Abfragesprache: Structured Query Language (SQL)

Literatur: z. B. [https://de.wikibooks.org/wiki/Einführung\\_in\\_SQL](https://de.wikibooks.org/wiki/Einführung_in_SQL)

Wichtige SQL-Befehle:

- **CREATE** – Datenbanken, Tabellen usw. anlegen
- **DROP** – Datenbanken, Tabellen usw. löschen
- **SELECT** – Daten abfragen  
auswählen mit **WHERE**, sortieren mit **ORDER BY**
- **INSERT INTO ... VALUES** – Daten eingeben
- **UPDATE** – Daten ändern
- **DELETE FROM** – Daten löschen

(Manche DBMS akzeptieren nur Großbuchstaben.)

Felder können auch **NULL** sein (ohne Inhalt).

# 4 Relationale Datenbanken

## 4.3 Normalformen

Problem: Schlecht angelegte Datenbanken werden schnell inkonsistent.  
Beliebte Fehler:

- Speichern von mehreren Daten in demselben Tabelleneintrag
- Speichern von denselben Daten in verschiedenen Tabelleneinträgen
- implizite Zusammenhänge
- voneinander unabhängige Zusammenhänge in derselben Tabelle

Literatur: z. B. [https://de.wikipedia.org/wiki/Normalisierung\\_\(Datenbank\)](https://de.wikipedia.org/wiki/Normalisierung_(Datenbank))

# 4 Relationale Datenbanken

## 4.3 Normalformen

Problem: Schlecht angelegte Datenbanken werden schnell inkonsistent.  
Beliebte Fehler:

- Speichern von mehreren Daten in demselben Tabelleneintrag  
→ 1. Normalform
- Speichern von denselben Daten in verschiedenen Tabelleneinträgen  
→ 2. Normalform
- implizite Zusammenhänge  
→ 3. Normalform und Boyce-Codd-Normalform
- voneinander unabhängige Zusammenhänge in derselben Tabelle  
→ 4. und 5. Normalform

Lösung: Normalformen

Literatur: z. B. [https://de.wikipedia.org/wiki/Normalisierung\\_\(Datenbank\)](https://de.wikipedia.org/wiki/Normalisierung_(Datenbank))

## 4 Relationale Datenbanken

### 4.4 Verknüpfungen von Tabellen

Problem: Gut angelegte Datenbanken (→ Normalformen) sind stark aufgesplittet.

Wie kann man sie weiterhin effizient benutzen?

Lösung: Verknüpfungen von Tabellen

SQL-Befehl: JOIN

Literatur: z. B. <https://de.wikipedia.org/wiki/SQL>



## 4 Relationale Datenbanken

### 4.4 Verknüpfungen von Tabellen

Was machen wir mit Tabelleneinträgen,  
bei denen die **ON**-Bedingung nicht erfüllt ist?

```
SELECT <Feld[er]> FROM <Tabelle1> [INNER] JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ weglassen: **[INNER] JOIN**

```
SELECT <Feld[er]> FROM <Tabelle1> LEFT JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ linke Tabelle trotzdem anzeigen (mit **NULL**-Einträgen): **LEFT JOIN**  
(analog: **RIGHT JOIN** für rechte Tabelle)

```
SELECT <Feld[er]> FROM <Tabelle1> FULL JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ beide Tabellen trotzdem anzeigen (mit **NULL**-Einträgen): **FULL JOIN**

## 4 Relationale Datenbanken

### 4.5 Sichten

```
SELECT <Feld[er]> FROM <Tabelle1> [INNER] JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ Wir betrachten beide Tabellen zusammen als eine große Tabelle.

```
CREATE VIEW <Sicht> AS  
    SELECT <Feld[er]> FROM <Tabelle1> JOIN <Tabelle2>  
        ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;  
SELECT <Feld[er]> FROM <Sicht> [WHERE ...];
```

→ Wir sprechen das Ergebnis genau wie eine Tabelle an.

→ Es ist möglich, ohne Verlust an Komfort alle Daten in Normalform zu halten.

# 4 Relationale Datenbanken

## 4.6 Schlüsselfelder

```
CREATE TABLE tabelle1 (  
    id INTEGER,  
    ...  
);  
CREATE TABLE tabelle2 (  
    ...  
    tabelle1_id INTEGER,  
    ...  
);  
ALTER TABLE tabelle1 ADD CONSTRAINT tabelle1_pkey  
    PRIMARY KEY (id);  
ALTER TABLE tabelle2 ADD CONSTRAINT tabelle2_fkey  
    FOREIGN KEY (tabelle1_id) REFERENCES tabelle1 (id);
```

- Dem DBMS mitteilen, welche Felder für **JOIN** vorgesehen sind.
- Das DBMS kann mit auf Konsistenz achten.

## 4 Relationale Datenbanken

### 4.6 Schlüsselfelder

- Dem DBMS mitteilen, welche Felder für **JOIN** vorgesehen sind.
- Das DBMS kann mit auf Konsistenz achten.

Beispiel: Man kann Daten, die noch referenziert werden, nicht löschen.

Alternative: Beim Löschen referenzierter Daten,  
referenzierende Daten mit löschen.

```
ALTER TABLE tabelle2 ADD CONSTRAINT tabelle2_fkey  
    FOREIGN KEY (tabelle1_id) REFERENCES tabelle1 (id)  
    ON DELETE CASCADE;
```

- Naher Verwandter eines Triggers (kommt gleich)

## 4 Relationale Datenbanken

### 4.7 Datensicherung

```
$ pg_dump --clean -h <Rechner> -U <User> -W <Datenbank>
```

- Ausgabe des gesamten Datenbankinhalts als SQL-Quelltext zur Standardausgabe

—> keine Probleme mit sich evtl. ändernden Binärformaten

- Es ist möglich, den Inhalt direkt in einer Pipe weiterzuverarbeiten (z. B. zu komprimieren).
- Zurückspielen: mit `psql`

```
$ psql -h <Rechner> -U <User> -W <Datenbank> \  
    < <Ausgabe von pg_dump>
```

- analog für **MariaDB**: `mariadb-dump`

# 4 Relationale Datenbanken

## 4.8 Aggregate

Rechenoperationen auf der ganzen Spalte, z. B. Summenbildung:

```
testdb=> SELECT SUM (id) FROM tier;
```

sum

-----

4143

Rechenoperationen: SUM, AVG, MIN, MAX, COUNT

Gruppierung:

```
testdb=> SELECT tierart, MIN (id) AS "Minimum" FROM tier
        GROUP BY tierart;
```

tierart	Minimum
Spinne	94
Hund	7
Kaninchen	42

Auswahl mit der anzuzeigenden Gruppen mit HAVING (entsprechend WHERE)

# 4 Relationale Datenbanken

## 4.9 Transaktionen

- Ziel: Wahrung der Konsistenz
- Methode: zusammengehörige Aktionen zu einer *Transaktion* zusammenfassen
- Beispiel: Überweisung  
Betrag von einem Konto subtrahieren  
und „gleichzeitig“ zu einem anderen Konto addieren
- Realisierung in PostgreSQL: `BEGIN; ... COMMIT;`
- Realisierung in MariaDB: `START TRANSACTION; ... COMMIT;`
- Abbruch einer Transaktion: `ROLLBACK;` statt `COMMIT;`

# 4 Relationale Datenbanken

## 4.10 Indizierung

- Ziel: Performanzgewinn
- Methode: Spalten, in denen häufig gesucht wird, sortieren
- Beispiel: Datenbank mit Kontaktdaten  
Suche nach Name, Adresse, Telefonnummer
- Realisierung in PostgreSQL und MariaDB:

```
CREATE INDEX <Indexname> ON <Tabellenname>
(
    <Spaltenname>,
    ...
);
```

- Vorteil: schnellere Suche  
(**SELECT**) in indizierten Spalten

Suche nach String:

Zeilen	ohne Index	mit Index
1000	1.809 ms	2.022 ms
1000000	501.546 ms	1.264 ms



# 4 Relationale Datenbanken

## 4.10 Indizierung

- Ziel: Performanzgewinn
- Methode: Spalten, in denen häufig gesucht wird, sortieren
- Beispiel: Datenbank mit Kontaktdaten  
Suche nach Name, Adresse, Telefonnummer
- Realisierung in PostgreSQL und MariaDB:

```
CREATE INDEX <Indexname> ON <Tabellenname>
(
    <Spaltenname>,
    ...
);
```

- Vorteil: schnellere Suche  
(**SELECT**) in indizierten Spalten
- Nachteil: langsames Einfügen/Ändern/Löschen

Suche nach String:

Zeilen	ohne Index	mit Index
1000	1.809 ms	2.022 ms
1000000	501.546 ms	1.264 ms

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

8. Januar 2025

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

- 1 Einführung
- 2 Kurzeinführung Unix
- 3 Kurzeinführung TCP/IP
- 4 Relationale Datenbanken
  - ...
  - 4.4 Verknüpfungen von Tabellen
  - 4.5 Sichten
  - 4.6 Schlüsselfelder
  - 4.7 Datensicherung
  - 4.8 Aggregate
  - 4.9 Transaktionen
  - 4.10 Indizierung
  - 4.11 Funktionen und Trigger
  - 4.12 GUI-Zugriff
  - 4.13 Datensicherheit bei Datenbanken
  - 4.14 Sonstige Datenbanken

...



Änderungen  
vorbehalten

## 4 Relationale Datenbanken

### 4.4 Verknüpfungen von Tabellen

Problem: Gut angelegte Datenbanken (→ Normalformen) sind stark aufgesplittet.

Wie kann man sie weiterhin effizient benutzen?

Lösung: Verknüpfungen von Tabellen

SQL-Befehl: JOIN

Literatur: z. B. <https://de.wikipedia.org/wiki/SQL>

## 4 Relationale Datenbanken

### 4.4 Verknüpfungen von Tabellen

Was machen wir mit Tabelleneinträgen,  
bei denen die **ON**-Bedingung nicht erfüllt ist?

```
SELECT <Feld[er]> FROM <Tabelle1> [INNER] JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ weglassen: **[INNER] JOIN**

```
SELECT <Feld[er]> FROM <Tabelle1> LEFT JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ linke Tabelle trotzdem anzeigen (mit **NULL**-Einträgen): **LEFT JOIN**  
(analog: **RIGHT JOIN** für rechte Tabelle)

```
SELECT <Feld[er]> FROM <Tabelle1> FULL JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ beide Tabellen trotzdem anzeigen (mit **NULL**-Einträgen): **FULL JOIN**

## 4 Relationale Datenbanken

### 4.5 Sichten

```
SELECT <Feld[er]> FROM <Tabelle1> [INNER] JOIN <Tabelle2>  
    ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;
```

→ Wir betrachten beide Tabellen zusammen als eine große Tabelle.

```
CREATE VIEW <Sicht> AS  
    SELECT <Feld[er]> FROM <Tabelle1> JOIN <Tabelle2>  
        ON <Tabelle1>.<Feld> = <Tabelle2>.<Feld>;  
SELECT <Feld[er]> FROM <Sicht> [WHERE ...];
```

→ Wir sprechen das Ergebnis genau wie eine Tabelle an.

→ Es ist möglich, ohne Verlust an Komfort alle Daten in Normalform zu halten.

# 4 Relationale Datenbanken

## 4.6 Schlüsselfelder

```
CREATE TABLE tabelle1 (  
    id INTEGER,  
    ...  
);  
CREATE TABLE tabelle2 (  
    ...  
    tabelle1_id INTEGER,  
    ...  
);  
ALTER TABLE tabelle1 ADD CONSTRAINT tabelle1_pkey  
    PRIMARY KEY (id);  
ALTER TABLE tabelle2 ADD CONSTRAINT tabelle2_fkey  
    FOREIGN KEY (tabelle1_id) REFERENCES tabelle1 (id);
```

- Dem DBMS mitteilen, welche Felder für **JOIN** vorgesehen sind.
- Das DBMS kann mit auf Konsistenz achten.

## 4 Relationale Datenbanken

### 4.6 Schlüsselfelder

- Dem DBMS mitteilen, welche Felder für **JOIN** vorgesehen sind.
- Das DBMS kann mit auf Konsistenz achten.

Beispiel: Man kann Daten, die noch referenziert werden, nicht löschen.

Alternative: Beim Löschen referenzierter Daten, referenzierende Daten mit löschen.

```
ALTER TABLE tabelle2 ADD CONSTRAINT tabelle2_fkey  
    FOREIGN KEY (tabelle1_id) REFERENCES tabelle1 (id)  
    ON DELETE CASCADE;
```

- Naher Verwandter eines Triggers (kommt gleich)



# 4 Relationale Datenbanken

## 4.7 Datensicherung

```
$ pg_dump --clean -h <Rechner> -U <User> -W <Datenbank>
```

- Ausgabe des gesamten Datenbankinhalts als SQL-Quelltext zur Standardausgabe

—> keine Probleme mit sich evtl. ändernden Binärformaten

- Es ist möglich, den Inhalt direkt in einer Pipe weiterzuverarbeiten (z. B. zu komprimieren).
- Zurückspielen: mit `psql`

```
$ psql -h <Rechner> -U <User> -W <Datenbank> \  
    < <Ausgabe von pg_dump>
```

- analog für **MariaDB**: `mariadb-dump`
- Alternative: Binär-Sicherung (auch zur Laufzeit möglich)  
—> Rücksichern schneller als mit Klartext-Sicherung

# 4 Relationale Datenbanken

## 4.8 Aggregate

Rechenoperationen auf der ganzen Spalte, z. B. Summenbildung:

```
testdb=> SELECT SUM (id) FROM tier;  
sum
```

```
-----  
4143
```

Rechenoperationen: SUM, AVG, MIN, MAX, COUNT

Gruppierung:

```
testdb=> SELECT tierart, MIN (id) AS "Minimum" FROM tier  
        GROUP BY tierart;
```

tierart	Minimum
Spinne	94
Hund	7
Kaninchen	42

Auswahl mit der anzuzeigenden Gruppen mit HAVING (entsprechend WHERE)

# 4 Relationale Datenbanken

## 4.9 Transaktionen

- Ziel: Wahrung der Konsistenz
- Methode: zusammengehörige Aktionen zu einer *Transaktion* zusammenfassen
- Beispiel: Überweisung  
Betrag von einem Konto subtrahieren  
und „gleichzeitig“ zu einem anderen Konto addieren
- Realisierung in PostgreSQL: `BEGIN; ... COMMIT;`
- Realisierung in MariaDB: `START TRANSACTION; ... COMMIT;`
- Abbruch einer Transaktion: `ROLLBACK;` statt `COMMIT;`

# 4 Relationale Datenbanken

## 4.10 Indizierung

- Ziel: Performanzgewinn
- Methode: Spalten, in denen häufig gesucht wird, sortieren
- Beispiel: Datenbank mit Kontaktdaten  
Suche nach Name, Adresse, Telefonnummer
- Realisierung in PostgreSQL und MariaDB:

```
CREATE INDEX <Indexname> ON <Tabellenname>
(
    <Spaltenname>,
    ...
);
```

- Vorteil: schnellere Suche  
(**SELECT**) in indizierten Spalten

Suche nach String:

Zeilen	ohne Index	mit Index
1000	1.809 ms	2.022 ms
1000000	501.546 ms	1.264 ms

# 4 Relationale Datenbanken

## 4.10 Indizierung

- Ziel: Performanzgewinn
- Methode: Spalten, in denen häufig gesucht wird, sortieren
- Beispiel: Datenbank mit Kontaktdaten  
Suche nach Name, Adresse, Telefonnummer
- Realisierung in PostgreSQL und MariaDB:

```
CREATE INDEX <Indexname> ON <Tabellenname>
(
    <Spaltenname>,
    ...
);
```

- Vorteil: schnellere Suche  
(**SELECT**) in indizierten Spalten
- Nachteil: langsames Einfügen/Ändern/Löschen

Suche nach String:

Zeilen	ohne Index	mit Index
1000	1.809 ms	2.022 ms
1000000	501.546 ms	1.264 ms

## 4 Relationale Datenbanken

### 4.11 Funktionen und Trigger

Funktionen:

- **PROCEDURE** entspricht einer **void**-Funktion in C.
- <https://www.postgresql.org/docs/15/sql-createprocedure.html>

Trigger:

- SQL-Standard: <https://www.sqltutorial.org/sql-triggers/>
- PostgreSQL: <https://www.postgresqltutorial.com/postgresql-triggers/creating-first-trigger-postgresql/>
- Anwendungsbeispiele:
  - Konsistenz
  - Protokollieren
  - Echtzeit-Aktualisierung bei DBMS-Migration

# 4 Relationale Datenbanken

## 4.12 GUI-Zugriff

- Anwendung nutzt DBMS-Client-Bibliothek  
GUI-Programmierung: wie gewohnt
- Spezialfall: Web-Anwendung

Beispiel: Programmiersprache PHP

- Integration in HTML-Quelltext: `<?php ... ?>`
- Objekt zur Kommunikation mit Datenbanken: PDO

Literatur:

- <https://www.postgresqltutorial.com/postgresql-php/connect/>
- <https://www.phptutorial.net/php-pdo/pdo-connecting-to-postgresql/>

## 4 Relationale Datenbanken

### 4.13 Datensicherheit bei Datenbanken

- kein direkter Zugriff von außen auf die Datenbank
- Transportverschlüsselung
- feingranulare Benutzerrechte
- Software aktuell halten
- gegen SQL Injection: Prepared Statements



## 4 Relationale Datenbanken

### 4.13 Datensicherheit bei Datenbanken: SQL Injection

Problem:

- Ein böswilliger Benutzer gibt über eine Benutzerschnittstelle (z. B. ein Web-Interface) Daten ein (z. B. einen „Namen“), die Sonderzeichen enthalten, damit sie als SQL-Befehle ausgeführt werden.
- Beispiele: <https://xkcd.com/327/>, <https://www.heise.de/-10220617>

Lösung: Die Benutzerschnittstelle prüft die Daten auf Sonderzeichen und ersetzt diese durch geeignete Escape-Sequenzen

- ' durch '' ersetzen
- Funktion `CHR ( )`
- Viele DBMS verstehen ein vorangestelltes \.

Bessere Lösung: *Prepared Statements*

- <https://www.postgresql.org/docs/current/sql-prepare.html>
- [https://www.w3schools.com/php/php\\_mysql\\_prepared\\_statements.asp](https://www.w3schools.com/php/php_mysql_prepared_statements.asp)

## 4 Relationale Datenbanken

### 4.14 Sonstige Datenbanken

- Eingebettete Datenbanken:  
Berkeley DB, SQLite  
Software-Bibliothek, keine Client-Server-Struktur
- Nicht-relationale Datenbanken:  
dokumentenorientierte Datenbanken, noSQL  
Performanz wichtiger als Konsistenz  
→ Applikationen stärker in Konsistenzprüfung eingebunden

# 5 Kryptographie

## 5.1 Einführung

### Was ist Datensicherheit?

(CIA)

- Vertraulichkeit (confidentiality) → Verschlüsselung
- Integrität (integrity) → Konsistenzprüfungen, Prüfwerte, Signaturen
- Verfügbarkeit (availability) → Backups, Ausfallsicherheit
  
- Identifizierbarkeit (Authentizität, Nichtabstreitbarkeit, Zurechenbarkeit)  
→ Passwörter, Signaturen  
bzw.
- Anonymität (plausible Abstreitbarkeit, Nichtzurechenbarkeit)  
→ Pseudonymisierung, Anonymisierung,  
Verschlüsselung, Steganographie

# 5 Kryptographie

## 5.1 Einführung

### Was ist Datensicherheit?

- (CIA)
- Vertraulichkeit (confidentiality) → Verschlüsselung
- Integrität (integrity) → Konsistenzprüfungen, Prüfwerte, Signaturen
- Verfügbarkeit (availability) → Backups, Ausfallsicherheit
- Identifizierbarkeit (Authentizität, Nichtabstreitbarkeit, Zurechenbarkeit)  
→ Passwörter, Signaturen  
bzw.
- Anonymität (plausible Abstreitbarkeit, Nichtzurechenbarkeit)  
→ Pseudonymisierung, Anonymisierung,  
Verschlüsselung, Steganographie → Kryptographie

# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: Cäsar-Chiffre, monoalphabetische Substitution

# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: Cäsar-Chiffre, monoalphabetische Substitution, One Time Pad

# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: Cäsar-Chiffre, monoalphabetische Substitution, One Time Pad
  
- Problem: Schlüsselaustausch

# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: Cäsar-Chiffre, monoalphabetische Substitution, One Time Pad, Pseudozufallszahlengenerator, Startwert als Schlüssel
- Problem: Schlüsselaustausch



# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: Cäsar-Chiffre, monoalphabetische Substitution, One Time Pad, *spezielle* Pseudozufallszahlengeneratoren, Startwert als Schlüssel: Enigma, DES, 3DES, RC4, IDEA, Blowfish, TwoFish, CAST, AES, ...
- Problem: Schlüsselaustausch

# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

**Unsicher!**

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: **Cäsar-Chiffre**, **monoalphabetische Substitution**, One Time Pad, *spezielle* Pseudozufallszahlengeneratoren, Startwert als Schlüssel: **Enigma**, **DES**, **3DES**, **RC4**, IDEA, Blowfish, TwoFish, CAST, AES, ...
- Problem: Schlüsselaustausch

# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

Unsicher!

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: **Cäsar-Chiffre**, **monoalphabetische Substitution**, One Time Pad, *spezielle* Pseudozufallszahlengeneratoren, Startwert als Schlüssel: **Enigma**, **DES**, **3DES**, **RC4**, IDEA, Blowfish, TwoFish, CAST, AES, ...
- Problem: Schlüsselaustausch
- Lösung: *asymmetrische Verschlüsselung*

## 5.3 Asymmetrische Verschlüsselung

- verschiedene Schlüssel zum Ver- und Entschlüsseln:  
öffentlicher und privater Schlüssel
- Prinzip: mathematische Operation,  
einfach durchzuführen, schwer rückgängig zu machen
- Beispiel:  $N = p \cdot q$  – einfacher als Primfaktorzerlegung von  $N \longrightarrow$  RSA  
 $73 \cdot 97 = 7081$ : geht notfalls noch im Kopf  
Primfaktorzerlegung von 7081: mindestens schriftlich, besser mit Rechner
- Beispiel:  $c = b^a$  – einfacher als  $a = \log_b c \longrightarrow$  Diffie-Hellman, ElGamal  
 $7^5 = 16807$ : geht notfalls noch im Kopf  
 $\log_7 16807$ : mindestens schriftlich, besser mit Rechner

→ Details: Algorithmen und Datenstrukturen

## 5.3 Asymmetrische Verschlüsselung

- verschiedene Schlüssel zum Ver- und Entschlüsseln:  
öffentlicher und privater Schlüssel
- Prinzip: mathematische Operation,  
einfach durchzuführen, schwer rückgängig zu machen
- Beispiel:  $N = p \cdot q$  – einfacher als Primfaktorzerlegung von  $N \longrightarrow$  RSA  
 $73 \cdot 97 = 7081$ : geht notfalls noch im Kopf  
Primfaktorzerlegung von 7081: mindestens schriftlich, besser mit Rechner
- Beispiel:  $c = b^a$  – einfacher als  $a = \log_b c \longrightarrow$  Diffie-Hellman, ElGamal  
 $7^5 = 16807$ : geht notfalls noch im Kopf  
 $\log_7 16807$ : mindestens schriftlich, besser mit Rechner

→ Details: Algorithmen und Datenstrukturen

- Nachteil: wesentlich aufwendiger und daher langsamer  
als symmetrische Verschlüsselung

→ *hybride Verschlüsselung*: nur Schlüsselaustausch asymmetrisch,  
eigentliche Verschlüsselung symmetrisch

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

15. Januar 2025

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

## 2 Kurzeinführung Unix

## 3 Kurzeinführung TCP/IP

## 4 Relationale Datenbanken

...

### 4.11 Funktionen und Trigger

### 4.12 GUI-Zugriff

### 4.13 Datensicherheit bei Datenbanken

### 4.14 Sonstige Datenbanken

## 5 Kryptographie

### 5.1 Einführung

### 5.2 Symmetrische Verschlüsselung

### 5.3 Asymmetrische und hybride Verschlüsselung

### 5.4 Signaturen

### 5.5 Authentifizierung

### 5.6 Quantencomputer

## 6 Datensicherheit



Änderungen  
vorbehalten

# 4 Relationale Datenbanken

## 4.11 Funktionen und Trigger

Funktionen:

- **PROCEDURE** entspricht einer **void**-Funktion in C.
- <https://www.postgresql.org/docs/15/sql-createprocedure.html>

Trigger:

- SQL-Standard: <https://www.sqltutorial.org/sql-triggers/>
- PostgreSQL: <https://www.postgresqltutorial.com/postgresql-triggers/creating-first-trigger-postgresql/>
- Anwendungsbeispiele:
  - Konsistenz
  - Protokollieren
  - Echtzeit-Aktualisierung bei DBMS-Migration



# 4 Relationale Datenbanken

## 4.12 GUI-Zugriff

- Anwendung nutzt DBMS-Client-Bibliothek  
GUI-Programmierung: wie gewohnt
- Spezialfall: Web-Anwendung

Beispiel: Programmiersprache PHP

- Integration in HTML-Quelltext: `<?php ... ?>`
- Objekt zur Kommunikation mit Datenbanken: PDO

Literatur:

- <https://www.postgresqltutorial.com/postgresql-php/connect/>
- <https://www.phptutorial.net/php-pdo/pdo-connecting-to-postgresql/>

## 4 Relationale Datenbanken

### 4.13 Datensicherheit bei Datenbanken

- kein direkter Zugriff von außen auf die Datenbank
- Transportverschlüsselung
- feingranulare Benutzerrechte
- Software aktuell halten
- gegen SQL Injection: Prepared Statements

## 4 Relationale Datenbanken

### 4.13 Datensicherheit bei Datenbanken: SQL Injection

Problem:

- Ein böswilliger Benutzer gibt über eine Benutzerschnittstelle (z. B. ein Web-Interface) Daten ein (z. B. einen „Namen“), die Sonderzeichen enthalten, damit sie als SQL-Befehle ausgeführt werden.
- Beispiele: <https://xkcd.com/327/>, <https://www.heise.de/-10220617>

Suboptimale Lösung: Die Benutzerschnittstelle prüft die Daten auf Sonderzeichen und ersetzt diese durch geeignete Escape-Sequenzen

- ' durch '' ersetzen
- Funktion `CHR ( )`
- Viele DBMS verstehen ein vorangestelltes \.

Bessere Lösung: *Prepared Statements*

- <https://www.postgresql.org/docs/current/sql-prepare.html>
- [https://www.w3schools.com/php/php\\_mysql\\_prepared\\_statements.asp](https://www.w3schools.com/php/php_mysql_prepared_statements.asp)

## 4 Relationale Datenbanken

### 4.14 Sonstige Datenbanken

- Eingebettete Datenbanken:  
Berkeley DB, SQLite  
Software-Bibliothek, keine Client-Server-Struktur
- Nicht-relationale Datenbanken:  
dokumentenorientierte Datenbanken, noSQL  
Performanz wichtiger als Konsistenz  
→ Applikationen stärker in Konsistenzprüfung eingebunden

# 5 Kryptographie

## 5.1 Einführung

### Was ist Datensicherheit?

(CIA)

- Vertraulichkeit (confidentiality) → Verschlüsselung
- Integrität (integrity) → Konsistenzprüfungen, Prüfwerte, Signaturen
- Verfügbarkeit (availability) → Backups, Ausfallsicherheit
  
- Identifizierbarkeit (Authentizität, Nichtabstreitbarkeit, Zurechenbarkeit)  
→ Passwörter, Signaturen  
bzw.
- Anonymität (plausible Abstreitbarkeit, Nichtzurechenbarkeit)  
→ Pseudonymisierung, Anonymisierung,  
Verschlüsselung, Steganographie

# 5 Kryptographie

## 5.1 Einführung

### Was ist Datensicherheit?

- (CIA)
- Vertraulichkeit (confidentiality) → Verschlüsselung
- Integrität (integrity) → Konsistenzprüfungen, Prüfwerte, Signaturen
- Verfügbarkeit (availability) → Backups, Ausfallsicherheit
- Identifizierbarkeit (Authentizität, Nichtabstreitbarkeit, Zurechenbarkeit)  
→ Passwörter, Signaturen  
bzw.
- Anonymität (plausible Abstreitbarkeit, Nichtzurechenbarkeit)  
→ Pseudonymisierung, Anonymisierung,  
Verschlüsselung, Steganographie → Kryptographie

# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: Cäsar-Chiffre, monoalphabetische Substitution

# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: Cäsar-Chiffre, monoalphabetische Substitution



# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: Cäsar-Chiffre, monoalphabetische Substitution

Unsicher!



# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: **Cäsar-Chiffre**, **monoalphabetische Substitution**, One Time Pad (beweisbar sicher)

**Unsicher!**



# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: **Cäsar-Chiffre**, **monoalphabetische Substitution**,  
One Time Pad (beweisbar sicher),  
**Pseudozufallszahlengenerator**, **Startwert als Schlüssel**

**Unsicher!**



# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

**Unsicher!**

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: **Cäsar-Chiffre**, **monoalphabetische Substitution**,  
One Time Pad (beweisbar sicher),  
*spezielle* Pseudozufallszahlengeneratoren, Startwert als Schlüssel:  
**Enigma**, **DES**, **3DES**, **RC4**, IDEA, Blowfish, TwoFish, CAST, AES, ...

# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

**Unsicher!**

- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: **Cäsar-Chiffre**, **monoalphabetische Substitution**, One Time Pad (beweisbar sicher), *spezielle* Pseudozufallszahlengeneratoren, Startwert als Schlüssel: **Enigma**, **DES**, **3DES**, **RC4**, IDEA, Blowfish, TwoFish, CAST, AES, ...
- Problem: Schlüsselaustausch

# 5 Kryptographie

## Kryptographie

- Verschlüsselung: symmetrisch, asymmetrisch, hybrid
- Hashes: Einwegfunktionen, Salt
- Signaturen, Zertifikate
- Schlüsselaustausch

## 5.2 Symmetrische Verschlüsselung

**Unsicher!**



- Derselbe Schlüssel zum Ver- und Entschlüsseln
- Beispiele: **Cäsar-Chiffre**, **monoalphabetische Substitution**, One Time Pad (beweisbar sicher), *spezielle* Pseudozufallszahlengeneratoren, Startwert als Schlüssel: **Enigma**, **DES**, **3DES**, **RC4**, IDEA, Blowfish, TwoFish, CAST, AES, ...
- Problem: Schlüsselaustausch
- Lösung: *asymmetrische Verschlüsselung*

## 5.3 Asymmetrische und hybride Verschlüsselung

- verschiedene Schlüssel zum Ver- und Entschlüsseln:  
öffentlicher und privater Schlüssel
- Prinzip: mathematische Operation,  
einfach durchzuführen, schwer rückgängig zu machen
- Beispiel:  $N = p \cdot q$  – einfacher als Primfaktorzerlegung von  $N \longrightarrow$  RSA  
 $73 \cdot 97 = 7081$ : geht notfalls noch im Kopf  
Primfaktorzerlegung von 7081: mindestens schriftlich, besser mit Rechner
- Beispiel:  $c = b^a$  – einfacher als  $a = \log_b c \longrightarrow$  Diffie-Hellman, ElGamal  
 $7^5 = 16807$ : geht notfalls noch im Kopf  
 $\log_7 16807$ : mindestens schriftlich, besser mit Rechner

→ Details: Algorithmen und Datenstrukturen

## 5.3 Asymmetrische und hybride Verschlüsselung

- verschiedene Schlüssel zum Ver- und Entschlüsseln:  
öffentlicher und privater Schlüssel
- Prinzip: mathematische Operation,  
einfach durchzuführen, schwer rückgängig zu machen
- Beispiel:  $N = p \cdot q$  – einfacher als Primfaktorzerlegung von  $N \longrightarrow$  RSA  
 $73 \cdot 97 = 7081$ : geht notfalls noch im Kopf  
Primfaktorzerlegung von 7081: mindestens schriftlich, besser mit Rechner
- Beispiel:  $c = b^a$  – einfacher als  $a = \log_b c \longrightarrow$  Diffie-Hellman, ElGamal  
 $7^5 = 16807$ : geht notfalls noch im Kopf  
 $\log_7 16807$ : mindestens schriftlich, besser mit Rechner

→ Details: Algorithmen und Datenstrukturen

- Nachteil: wesentlich aufwendiger und daher langsamer  
als symmetrische Verschlüsselung

→ *hybride Verschlüsselung*: nur Schlüsselaustausch asymmetrisch,  
eigentliche Verschlüsselung symmetrisch



## 5.4 Signaturen

- *kryptographische Hash-Funktion*: leicht auszurechnen, schwer zu manipulieren
- asymmetrisch: *Signatur*  
Hash-Wert mit privatem Schlüssel verschlüsseln,  
mit öffentlichem Schlüssel entschlüsseln
- symmetrisch: *Message Authentication Code* (MAC)  
z. B. Hash-Wert über Nachricht + geheimer Schlüssel

## 5.4 Signaturen

- *kryptographische Hash-Funktion*: leicht auszurechnen, schwer zu manipulieren
- asymmetrisch: *Signatur*  
Hash-Wert mit privatem Schlüssel verschlüsseln,  
mit öffentlichem Schlüssel entschlüsseln
- symmetrisch: *Message Authentication Code* (MAC)  
z. B. Hash-Wert über Nachricht + geheimer Schlüssel

Angriffsmöglichkeit: *Man-in-the-middle*-Angriff

Beim Schlüsselaustausch anderen Schlüssel unterchieben

→ Sorgfalt beim Schlüsselaustausch

## 5.4 Signaturen

- *kryptographische Hash-Funktion*: leicht auszurechnen, schwer zu manipulieren
- asymmetrisch: *Signatur*  
Hash-Wert mit privatem Schlüssel verschlüsseln,  
mit öffentlichem Schlüssel entschlüsseln
- symmetrisch: *Message Authentication Code* (MAC)  
z. B. Hash-Wert über Nachricht + geheimer Schlüssel

Angriffsmöglichkeit: *Man-in-the-middle*-Angriff

Beim Schlüsselaustausch anderen Schlüssel unterchieben

→ Sorgfalt beim Schlüsselaustausch

### Praxis-Beispiele

- SSH
- HTTPS
- OpenPGP → Praktikumsversuch 3

## 5.5 Authentifizierung

- Zugangsdaten:  
Benutzername, Passwort
- Problem:  
Zugangsdaten mitlesen
- Lösung:  
verschlüsselte Verbindung
- Problem:  
Zugangsdaten speichern
- Lösung:  
Hash-Wert statt Passwort speichern
- Problem:  
gleiche Passwörter identifizierbar
- Lösung:  
*Salt*

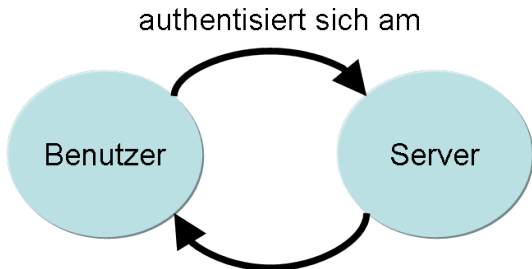


Bild: <https://de.wikipedia.org/wiki/Datei:Authentisieren-authentifizieren.png>

Hash-Algorithmen:

CRC → *kein* Hash-Algorithmus!

DES – unsicher (Unix traditionell)

MD5 – unsicher

SHA – Stand der Technik

weitere Algorithmen:

siehe *man 5 crypt*

## 5.5 Authentifizierung

- Zugangsdaten:  
Benutzername, Passwort
- Problem:  
Zugangsdaten mitlesen
- Lösung:  
verschlüsselte Verbindung
- Problem:  
Zugangsdaten speichern
- Lösung:  
*Challenge-Response-Authentifizierung*  
Beispiel: HTTP Digest

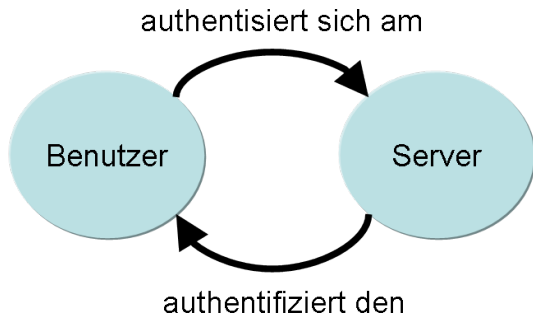


Bild: <https://de.wikipedia.org/wiki/Datei:Authentisieren-authentifizieren.png>

- gemeinsamer geheimer Schlüssel (Passwort)
- Server schickt *Nonce* an Benutzer
- Benutzer schickt Hash über [Passwort + Nonce] an Server
- Server berechnet denselben Hash → Authentifizierung erfolgreich
- Nonce (= „number used once“) nur einmal verwenden!

## 5.6 Quantencomputer

Analogie: Spaghetti-Sort

[https://en.wikipedia.org/wiki/Spaghetti\\_sort](https://en.wikipedia.org/wiki/Spaghetti_sort)

- Prinzip:  $2^n$  Berechnungen gleichzeitig  
( $n$  = Registerbreite)

—→ Klassisch schwierige Probleme werden einfacher.

- Beispiel: Primfaktorzerlegung:  $\mathcal{O}(n^3)$  statt  $\mathcal{O}(2^{\sqrt{n \log n}})$
- Problem für asymmetrische Verschlüsselungsalgorithmen,  
z. B. RSA (Primfaktorzerlegung), ElGamal (diskreter Logarithmus)
- weniger problematisch für symmetrische Verschlüsselungsalgorithmen

—→ *Post-Quanten-Kryptographie*

Beispiel: McEliece-Kryptosystem

—→ *Forward Secrecy*

Kompromittierung betrifft nur zukünftige Kommunikation,  
nicht bereits vergangene. Beispiel:

- RSA nur für Authentifizierung
- Austausch eines Sitzungsschlüssels via Diffie-Hellman
- Kommunikation über Sitzungsschlüssel (symmetrisch)
- Sitzungsschlüssel nur einmal verwenden!

# 6 Datensicherheit

## 6.1 Prinzipien: Wie erreichen wir Datensicherheit?

### Komplexitätsreduzierung

- Teilsysteme gegeneinander abgrenzen, Abhängigkeiten reduzieren
- fehlertolerantes Design

### Realistische Zeitplanung

- ... anstelle von:  
Probleme lange vor sich her schieben, dann alles auf einmal einfordern („gleichzeitig zu langsam und zu schnell“ – Martin Tschirsich, 2025)
- Problem: wirtschaftliche oder politische Zeitvorgaben
- Ziele nicht nachträglich ändern, sondern frühzeitig festlegen

### Verantwortungen klären und ausüben

- Beteiligten, die technisch Ahnung haben, Zeit, Freiraum und Autorität geben

### Transparenz

- Prozesse dokumentieren, auch Entwicklungsprozesse, offene Quelltexte

### Positivbeispiel: Corona-Warn-App

# 6 Datensicherheit

## 6.2 Netzwerksicherheit

- Firewall: nur bestimmte IP-Adressen / Ports / Inhalte zulassen
- VPN: verschlüsselte Verbindung von Netzwerken über ansonsten unsichere Verbindung (Internet)
- *Intrusion Detection System*

### Anonymität

- Beispiel: Tor – Zwiebel-Routing
  - Tor-Browser
  - Tails
- Beispiel: Corona-Warn-App

### Cross-Site-Scripting verhindern



# 6 Datensicherheit

## 6.2 Netzwerksicherheit

Die menschliche Komponente

- Bequemlichkeit
- Social Engineering
  - *Phishing*
  - KI-Sprachmodelle

*Es gibt für jedes menschliche Problem  
immer eine wohl bekannte Lösung -  
sauber, einleuchtend, und falsch.*

Henry Louis Mencken, The Divine Afflatus, 1917  
<https://de.wikiquote.org/wiki/Lösung>

# 6 Datensicherheit

## 6.3 Verfügbarkeit

Wann wird wirklich auf den Datentäger geschrieben?

- DBMS: Persistenz-Einstellungen
- *Write Ahead Log* (WAL)  
*Journaling-Dateisysteme*
- *CAP-Theorem*

Daten sicher aufbewahren

- Backup
- RAID

Hochverfügbarkeit

- allgemein: *High-Availability-Cluster*
- speziell: Datenbank-Cluster: Replikation über mehrere Server

# 6 Datensicherheit

## 6.4 Datenschutz

- Schutz vor mißbräuchlicher Datenverarbeitung
- informationelle Selbstbestimmung
- Persönlichkeitsrecht
- Privatsphäre

Datenmißbrauch ermöglicht hohe Gewinne

- viel Interesse an persönlichen Daten
- „Datenschutz hemmt ~~den Fortschritt!~~“ die persönliche Bereicherung

- DSGVO

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

## 2 Kurzeinführung Unix

## 3 Kurzeinführung TCP/IP

## 4 Relationale Datenbanken

## 5 Kryptographie

### 5.1 Einführung

### 5.2 Symmetrische Verschlüsselung

### 5.3 Asymmetrische und hybride Verschlüsselung

### 5.4 Signaturen

### 5.5 Authentifizierung

### 5.6 Quantencomputer

## 6 Datensicherheit

### 6.1 Prinzipien: Wie erreichen wir Datensicherheit?

### 6.2 Netzwerksicherheit

### 6.3 Verfügbarkeit

### 6.4 Datenschutz

***Vielen Dank für Ihre Aufmerksamkeit  
und viel Erfolg bei den Prüfungen!***